

# **Parallel I/O on Highly Parallel Systems**

## **Supercomputing '94 — Tutorial M11 Notes**

Bill Nitzberg<sup>1</sup> and Samuel A. Fineberg<sup>1</sup>

Report NAS-94-005 November 14, 1994

NAS Systems Division  
NASA Ames Research Center  
Mail Stop 258-6  
Moffett Field, CA 94035-1000

[nitzberg,fineberg]@nas.nasa.gov

### **Abstract**

Typical scientific applications require vast amounts of processing power coupled with significant I/O capacity. Highly parallel computer systems provide floating point processing power at low cost, but efficiently supporting a scientific workload also requires commensurate I/O performance. In order to achieve high I/O performance, these systems utilize parallelism in their I/O subsystems---supporting concurrent access to files by multiple nodes of a parallel application, and striping files across multiple disks. However, obtaining maximum I/O performance can require significant programming effort.

This tutorial presents a snapshot of the state of I/O on highly parallel systems by comparing the well-balanced I/O performance of a traditional vector supercomputer (the Cray Y/MP C90) with the I/O performance of various highly parallel systems (Cray T3D, IBM SP-2, Intel iPSC/860 and Paragon, and Thinking Machines CM-5). In addition, the tutorial covers benchmarking techniques for evaluating current parallel I/O systems and techniques for improving parallel I/O performance. Finally, the tutorial presents several high level parallel I/O libraries and shows how they can help application programmers improve I/O performance.

---

1. Computer Sciences Corporation, NASA Contract NAS 2-12961, Moffett Field, CA 94035-1000



# **Parallel I/O on Highly Parallel Systems: A Tutorial**

**Sam Fineberg  
and  
Bill Nitzberg**

**Computer Sciences Corporation  
NASA Ames Research Center  
Moffett Field, CA 94035-1000**

**November 14, 1994**

## **Parallel I/O Outline**

- Motivation—What is Parallel I/O?
- Current I/O Architectures
- Case Studies
- I/O Benchmarks
- I/O Libraries and User Interfaces
- Futures

## Slide Credits (Thanks!)

- Intel Supercomputer Systems Division, Brad Rullman (PFS), and Thanh Phung, Ed Kushner, Dave Minturn (NAS Application I/O Benchmark)
- MasPar Computer Corporation, John Nickolls
- Thinking Machines Corporation, Susan LoVerso, Marshall Isman, Andy Nanopoulos, Bill Nesheim, Ewan Milne, Rick Wheeler
- IBM T J Watson Research Center, Peter Corbett, Dror Feitelson, Sandra Baylor, Jean-Pierre Prost, Yarsun Hsu
- Subhash Saini, Horst Simon, NAS/CSC (SC '94)

## Also thanks to:

- Karen Allen, Rebecca Arney, Cray Research (Cray T3D details)
- Suga Sugavanam, Vijay Naik, Javed Khan, IBM, (IBM SP2 Application I/O Benchmark results)
- John George, Convex Computer Corporation

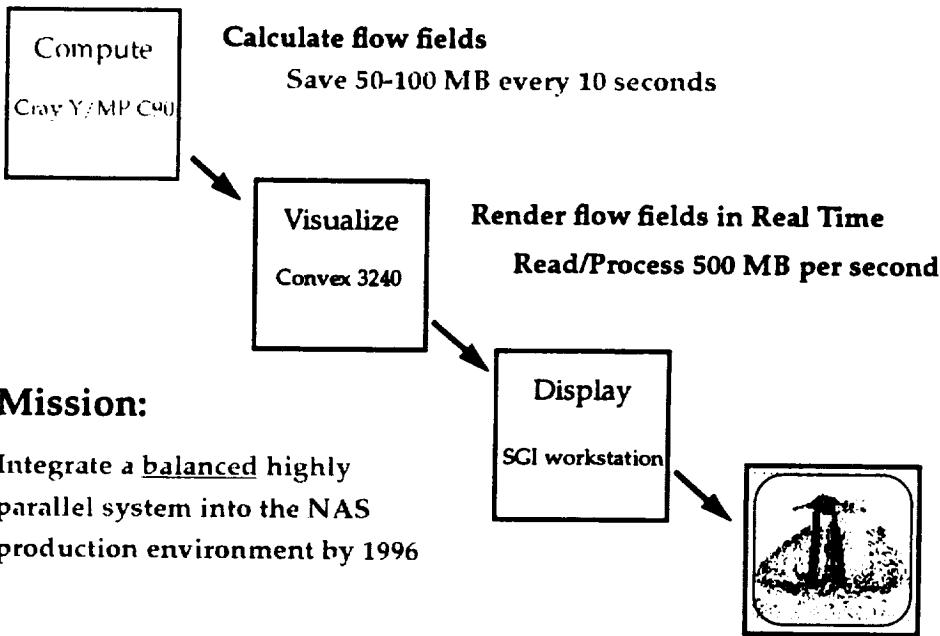
Copies of these tutorial materials can be found at URL:

<http://www.nas.nasa.gov/home.html>

or by sending email to '[doc-center@nas.nasa.gov](mailto:doc-center@nas.nasa.gov)'.



# Processing Model at NAS



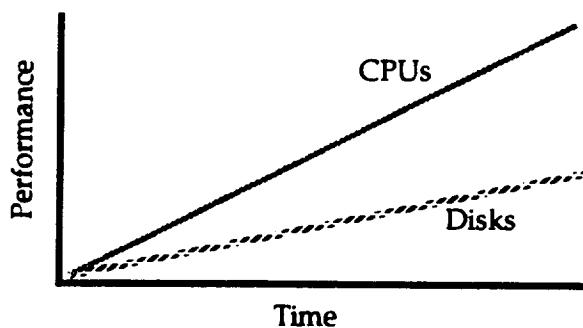
SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

## MPPs: "It's the economy, stupid."

Commodity "off-the-shelf" parts have made MPPs and workstation clusters the price/performance winners.

Processor speed is improving faster than disk speed.



The price is right; the performance is...

...only a matter of software.

SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

# What is "Parallel I/O"?

"Single" parallel application run on many nodes

Data is distributed among these nodes.

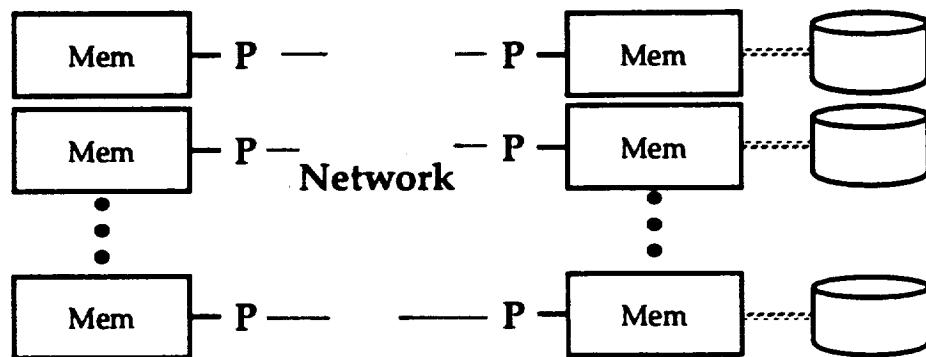
This data is moved to/from a single (logical) file.

The file is itself spread across nodes and disks.

## What about:

- RAIDs?
- Transaction Processing?
- Striped HiPPIs?

# Highly Parallel I/O



# Classes of Parallel I/O

## Application I/O

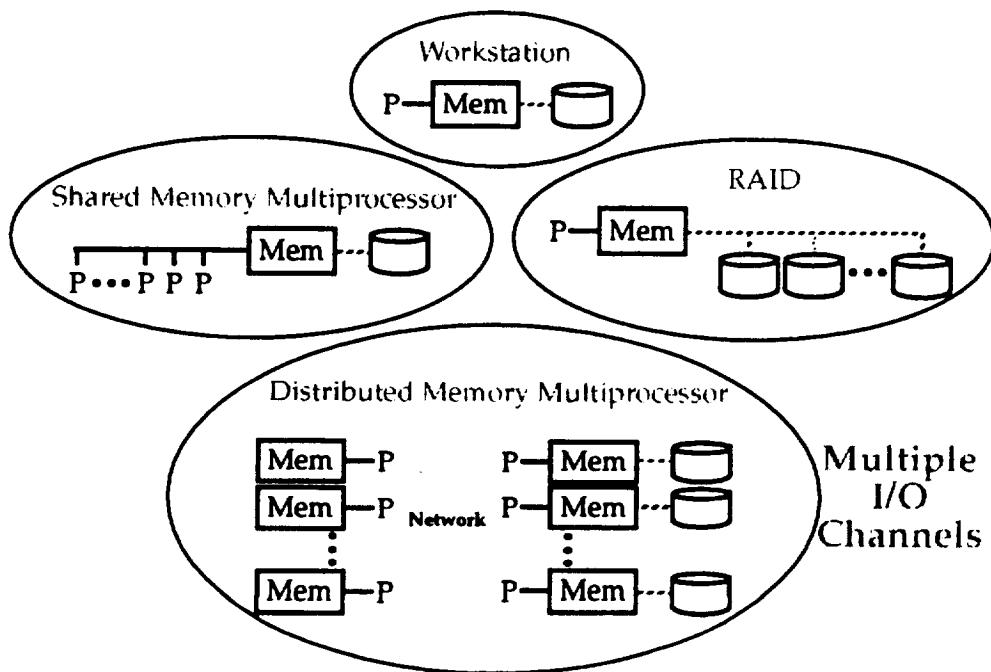
- Program Initialization — read only, once per run
- Data Analysis — read mostly “serial” files
- Visualization/Trace — write only “serial” files
- Out-of-core solvers — read/write “parallel” files
- Checkpointing — write mostly “parallel (?)” files

## System I/O

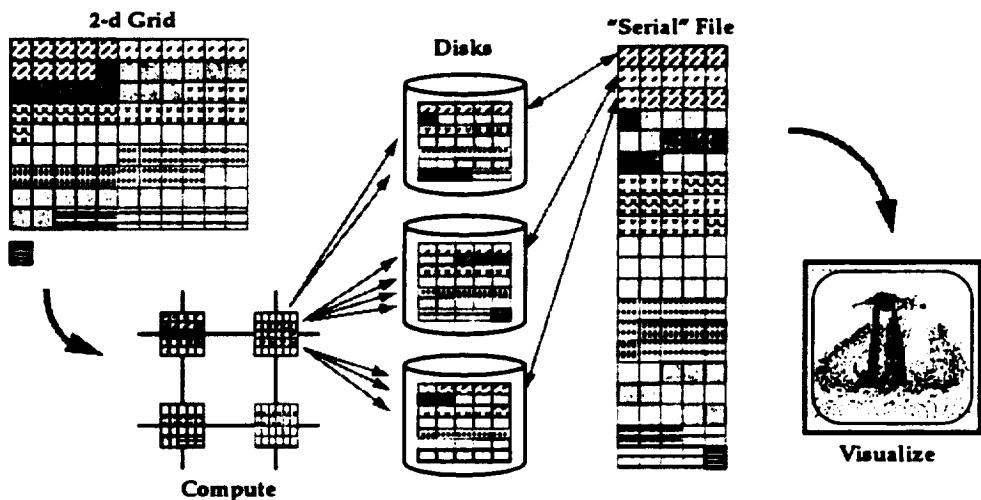
- Program Loading, Paging, Checkpointing
- Data Migration

Supporting a workload...

## What Makes Parallel I/O Hard?



# What (Else) Makes Parallel I/O Hard?



Data layout optimized for processors, not disks.

# Current I/O Architectures

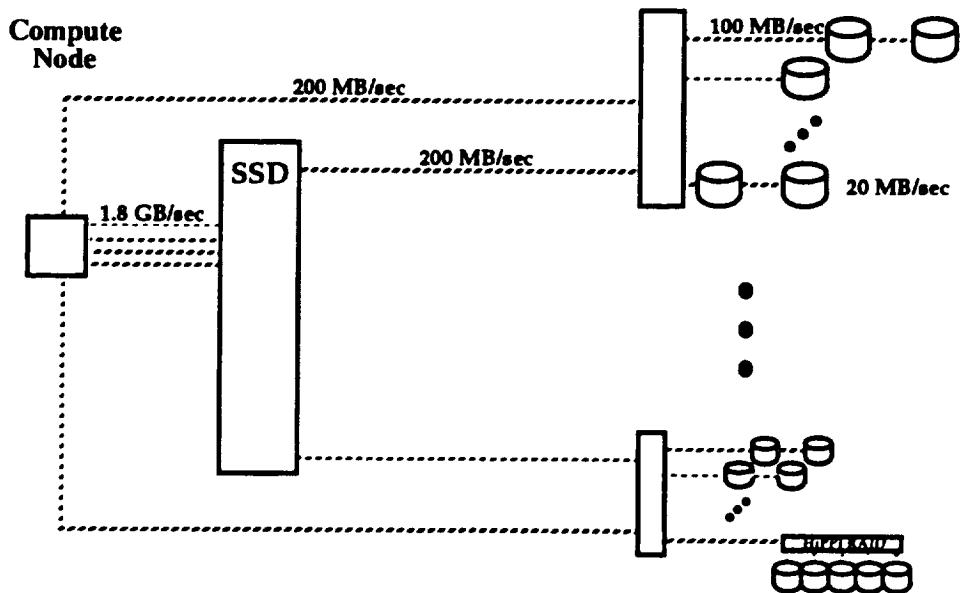
## "Serial" Systems

- Cray C90

## Parallel Systems

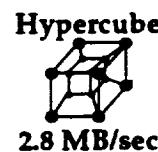
- Intel iPSC/860 (CFS—historical perspective)
- TMC CM-5 (SDA—SIMD)
- Intel Paragon (PFS—MIMD)
- IBM SP2 (workstation clusters)
- Cray T3D (front-end based)
- MasPar MP-2 (pure SIMD)

## Cray Research C90 I/O Architecture

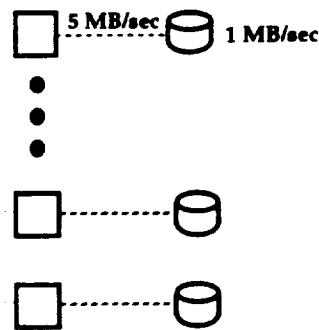


# Intel iPSC/860 I/O Architecture

Compute  
Nodes



I/O  
Nodes

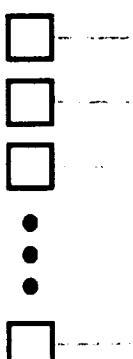


SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

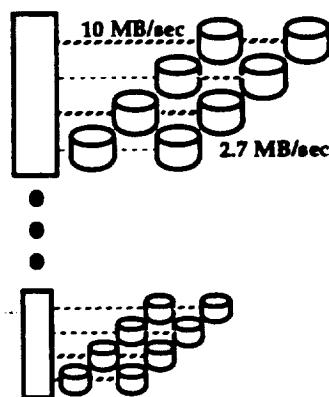
Nov. 14, 1994

# TMC CM-5 I/O Architecture

Compute  
Nodes



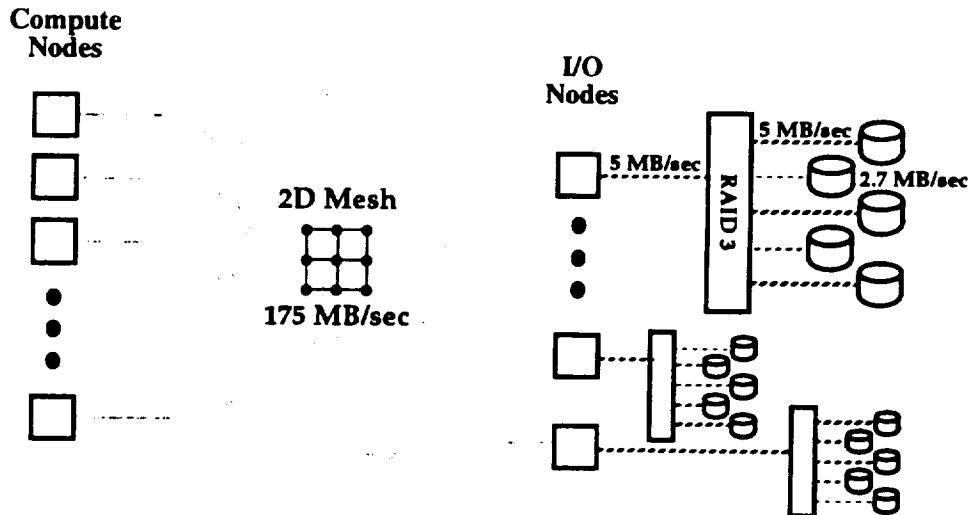
I/O  
Nodes



SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

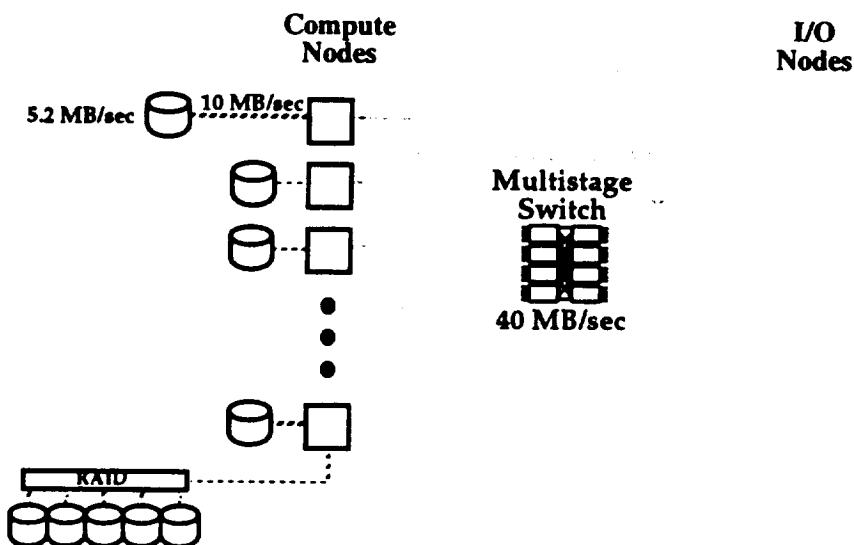
# Intel Paragon I/O Architecture



SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

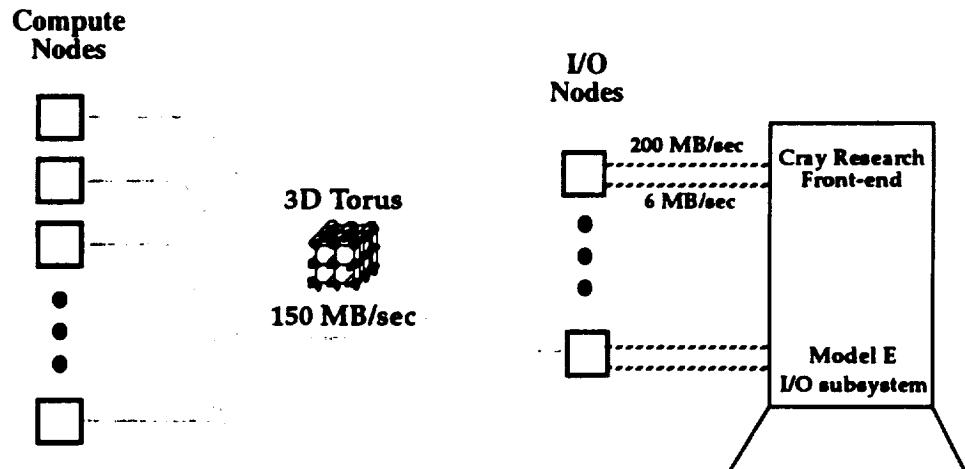
# IBM SP2 I/O Architecture



SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

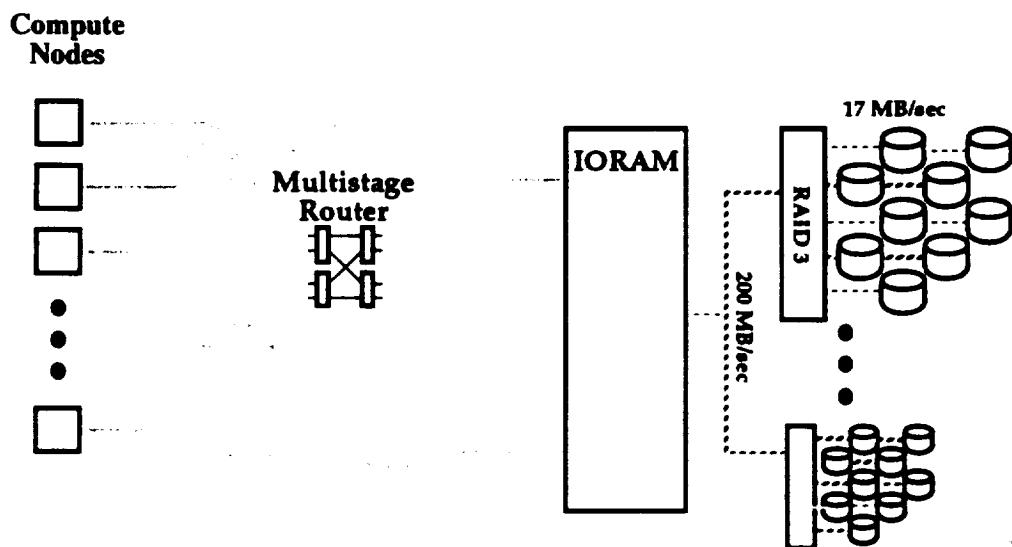
# Cray T3D I/O Architecture



SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

# MasPar MP-2 I/O Architecture



SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

# Case Studies

- Intel iPSC/860 (CFS)
- Thinking Machines Corp. CM-5E (sfs)
- Intel Paragon (PFS)
- IBM SP2
- Cray Research T3D
- MasPar MP-2



# Highly Parallel I/O Architecture

## Filesystem

- Model — Centralized vs. SIMD vs. MIMD
- Caching (size, location, hardware support)
- File layout (static vs. run time, block size, striping)

## Hardware

- I/O nodes (type, placement)
- Reliability — software vs. hardware RAIDs

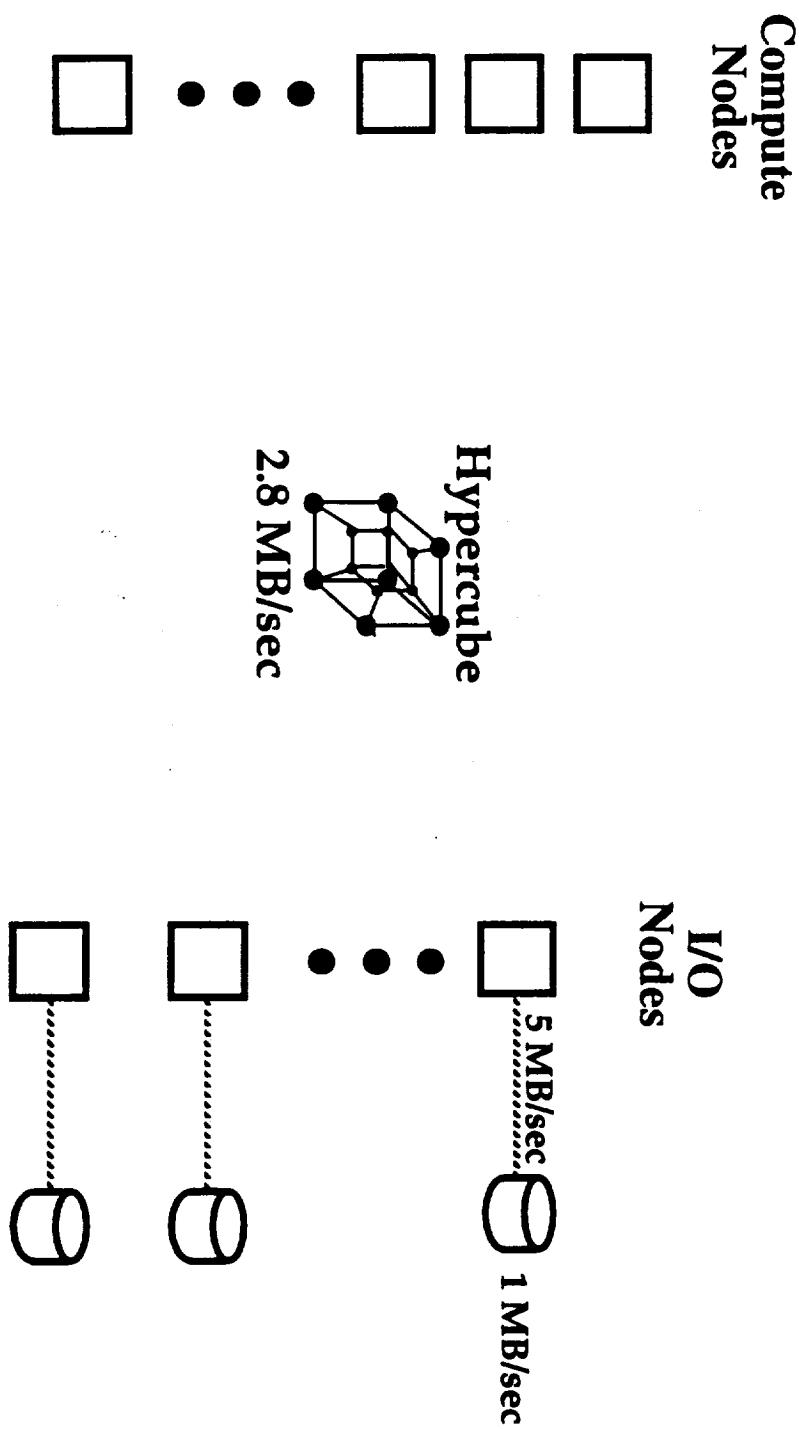
## Performance

- Scalability and Bottlenecks

## System Balance



# Intel iPSC/860 I/O Architecture



# **Performance Characteristics of the iPSC/860 and CM-2 I/O Systems**

**Bill Nitzberg  
and  
John Krystynak**

**Computer Sciences Corporation  
NASA Ames Research Center  
Moffett Field, CA 94035-1000**

**April 16, 1993**

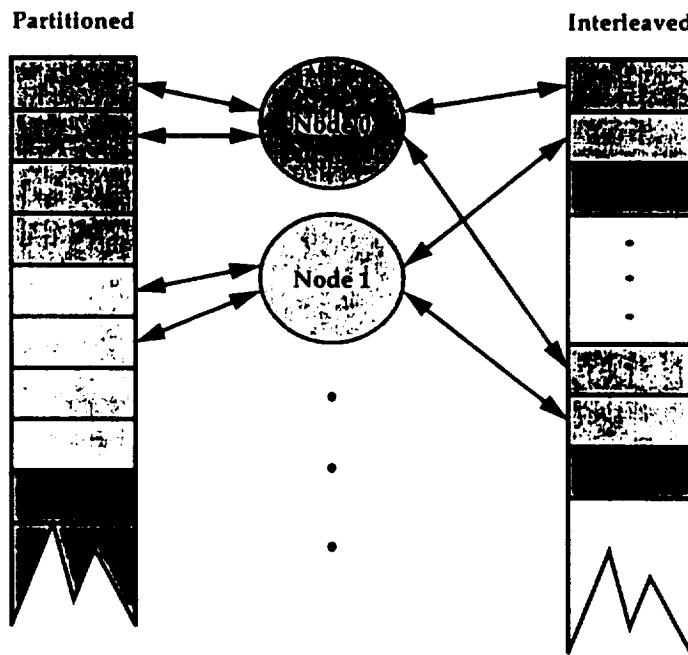


## **iPSC/860 Performance Tests**

- C array (repeatedly) transferred on each processor
- Partitioned vs. Interleaved file structure
- Runs for 1, 2, 4, 8, 16, 32, 64, and 128 processors
- File size for all tests was 128 MB
- Transfer sizes from 4KB to 1 MB
- Measurements reflect best of 3 trials on a dedicated system
- iPSC/860 System Software Version 3.3.1



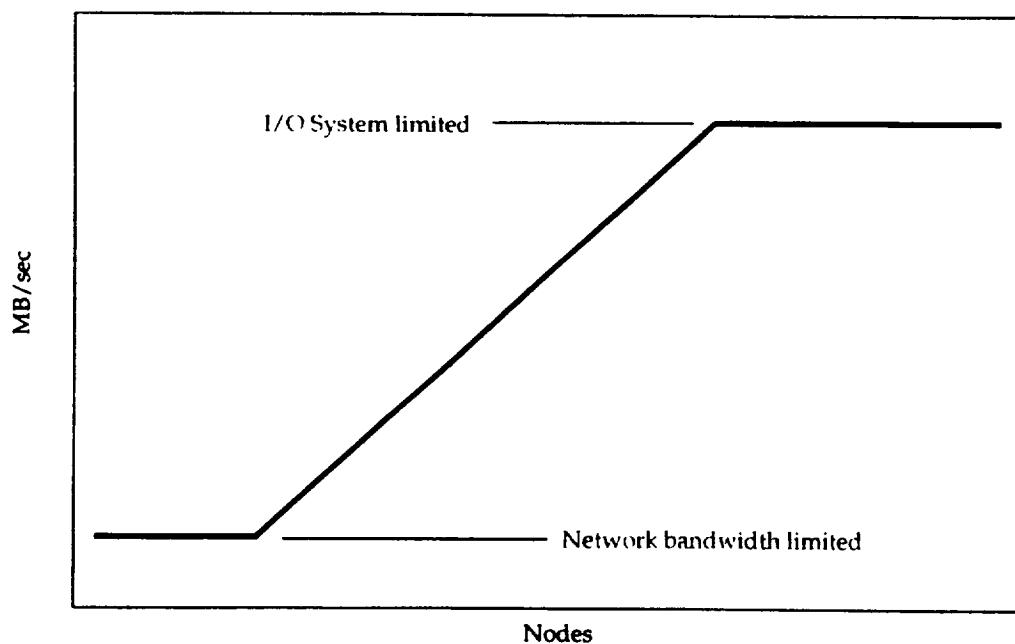
# Partitioned vs. Interleaved



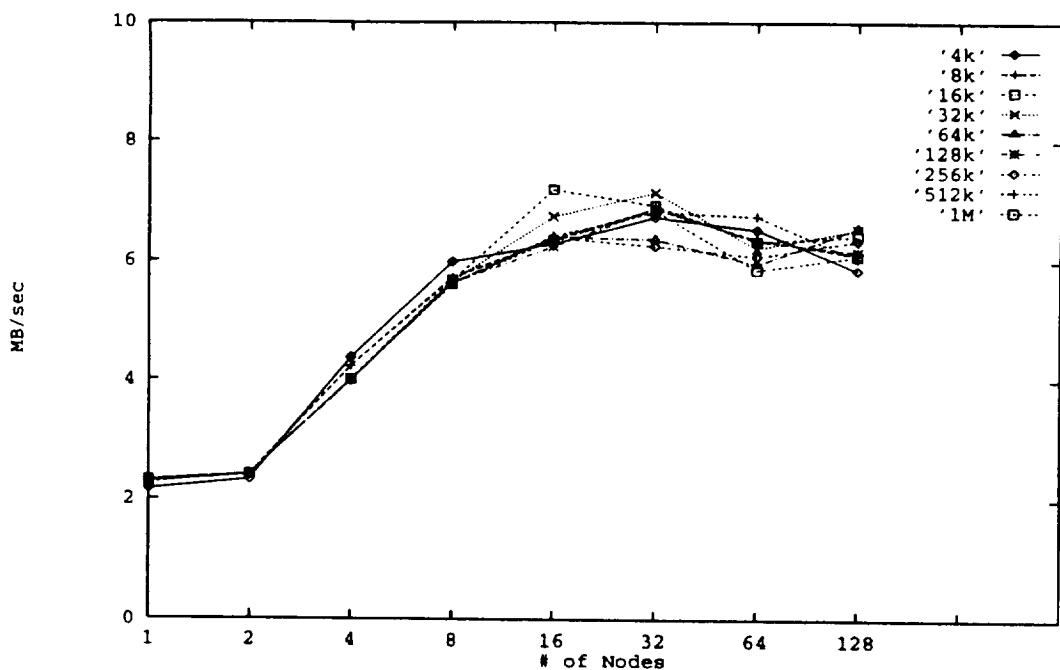
IPPS '93, "iPSC/860 and CM-2 I/O Systems", J. Krystynak and B. Nitzberg

April 16, 1993, p. 13

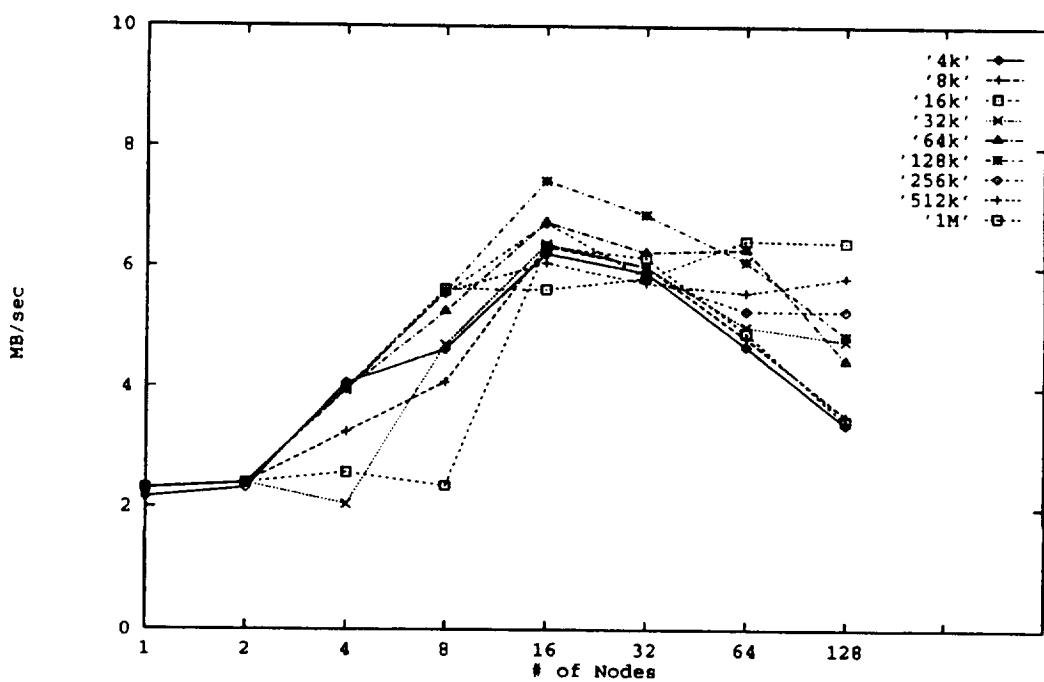
## Ideal Performance Curve



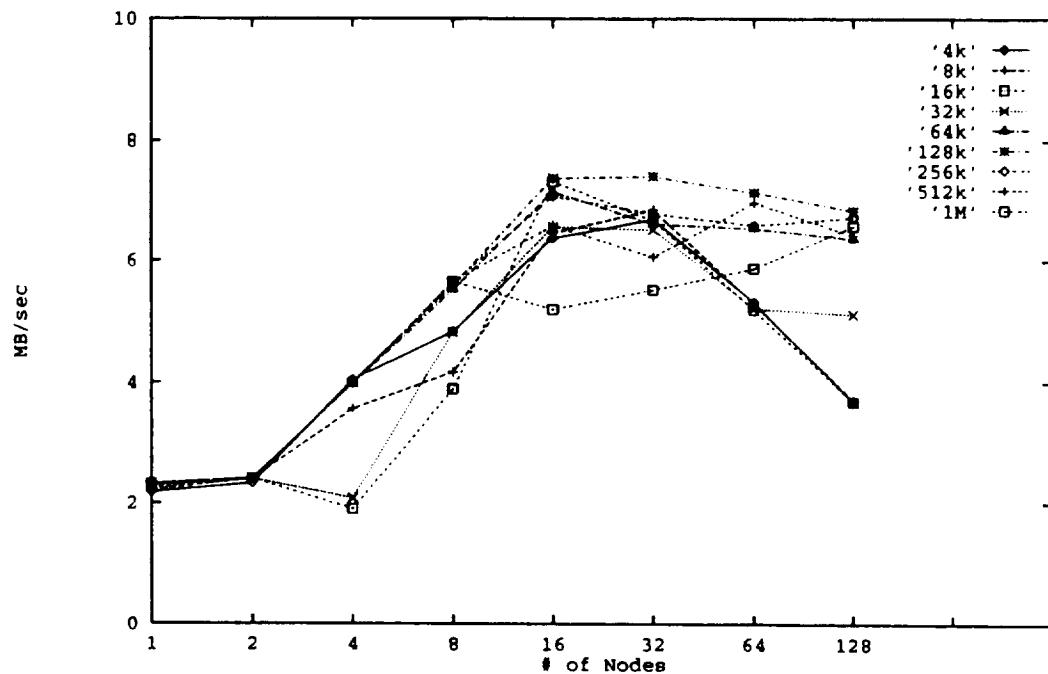
## iPSC/860 Write (Partitioned)



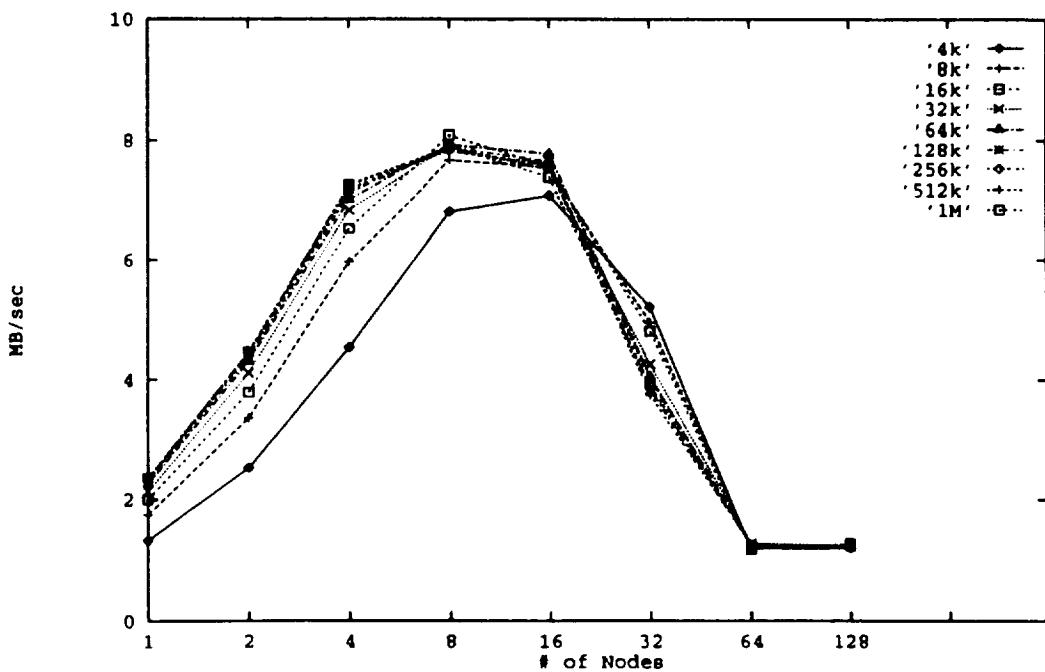
## iPSC/860 Write (Interleaved)



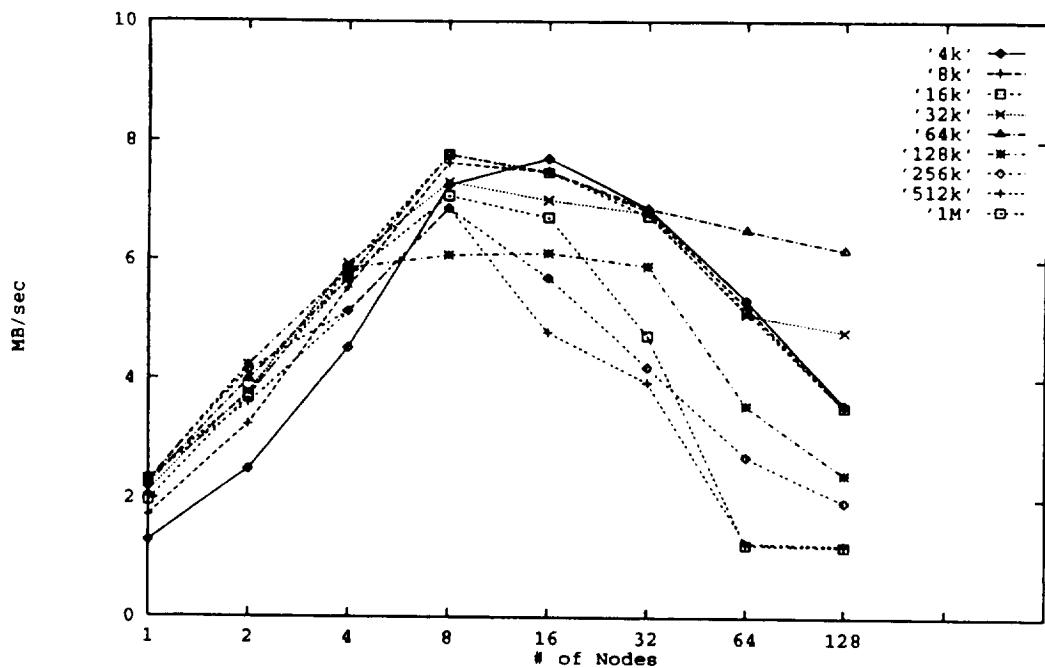
## iPSC/860 Write (Interleaved/Grouped)



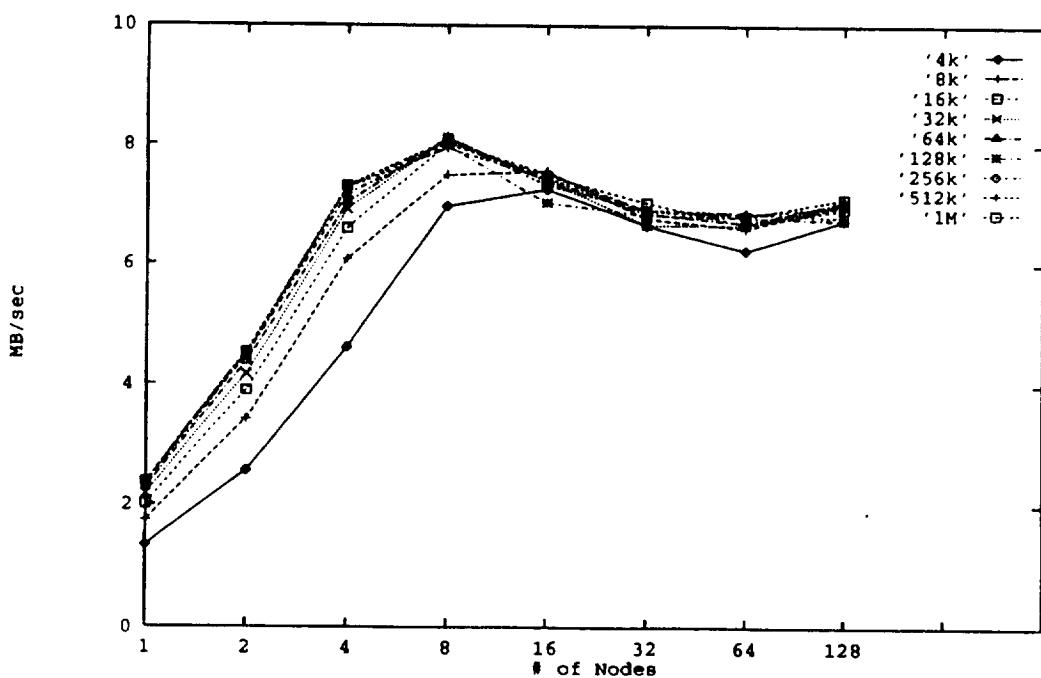
## iPSC/860 Read (Partitioned)



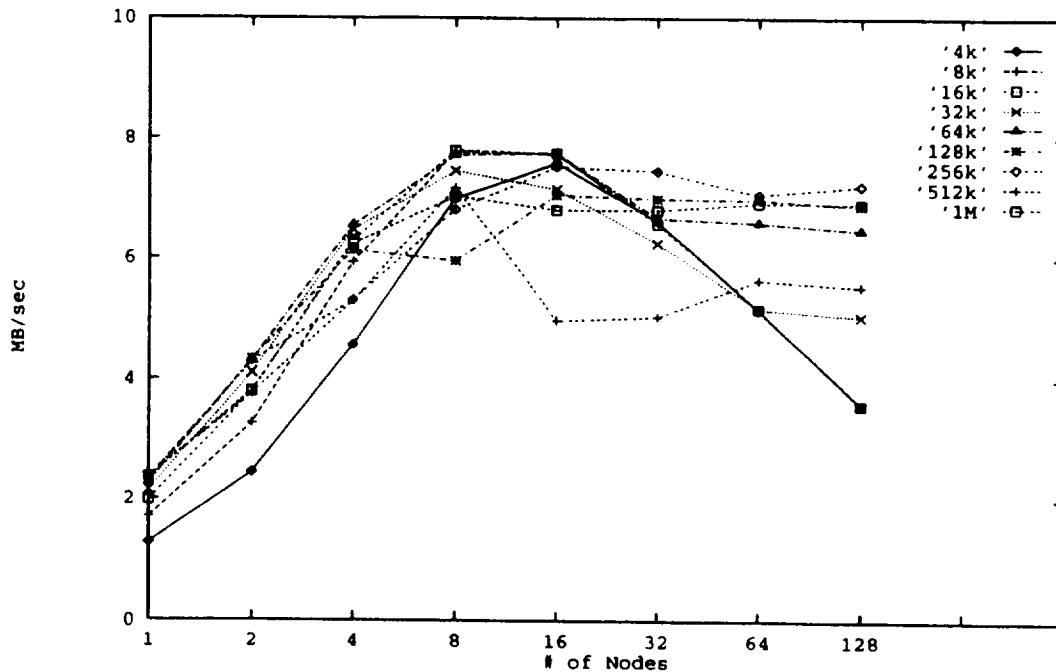
## iPSC/860 Read (Interleaved)



## iPSC/860 Read (Partitioned/Grouped)



# iPSC/860 Read (Interleaved/Grouped)



IPPS '93, "iPSC/860 and CM-2 I/O Systems", J. Krystynak and B. Nitzberg

April 16, 1993, p. 21

## Summary

### CM-2

- Sustains 50-80% of peak (15-25 MB/sec)
- Limitation: inability to use both CMIO buses simultaneously

### iPSC/860

- Sustains 70-80% of peak writing (7-8 MB/sec)
- Special tactics (e.g. grouping) required for read performance
- Undersized I/O subsystem for NAS iPSC/860

To order technical reports: doc-center@nas.nasa.gov

# iPSC/860 Summary

## File System: CFS

- MIMD, proprietary, UNIX file interface
- File blocks striped across I/O nodes
- 1 MB disk block cache per I/O node
- File blocks allocated dynamically

## I/O nodes:

- "Slow" compute nodes — Intel 386 w/ 4 MB
- Attached outside hypercube (2.8 MB/sec)
- 1 SCSI Maxtor 760 MB disk (1 MB/sec)

# iPSC/860 Summary, cont.

## Performance

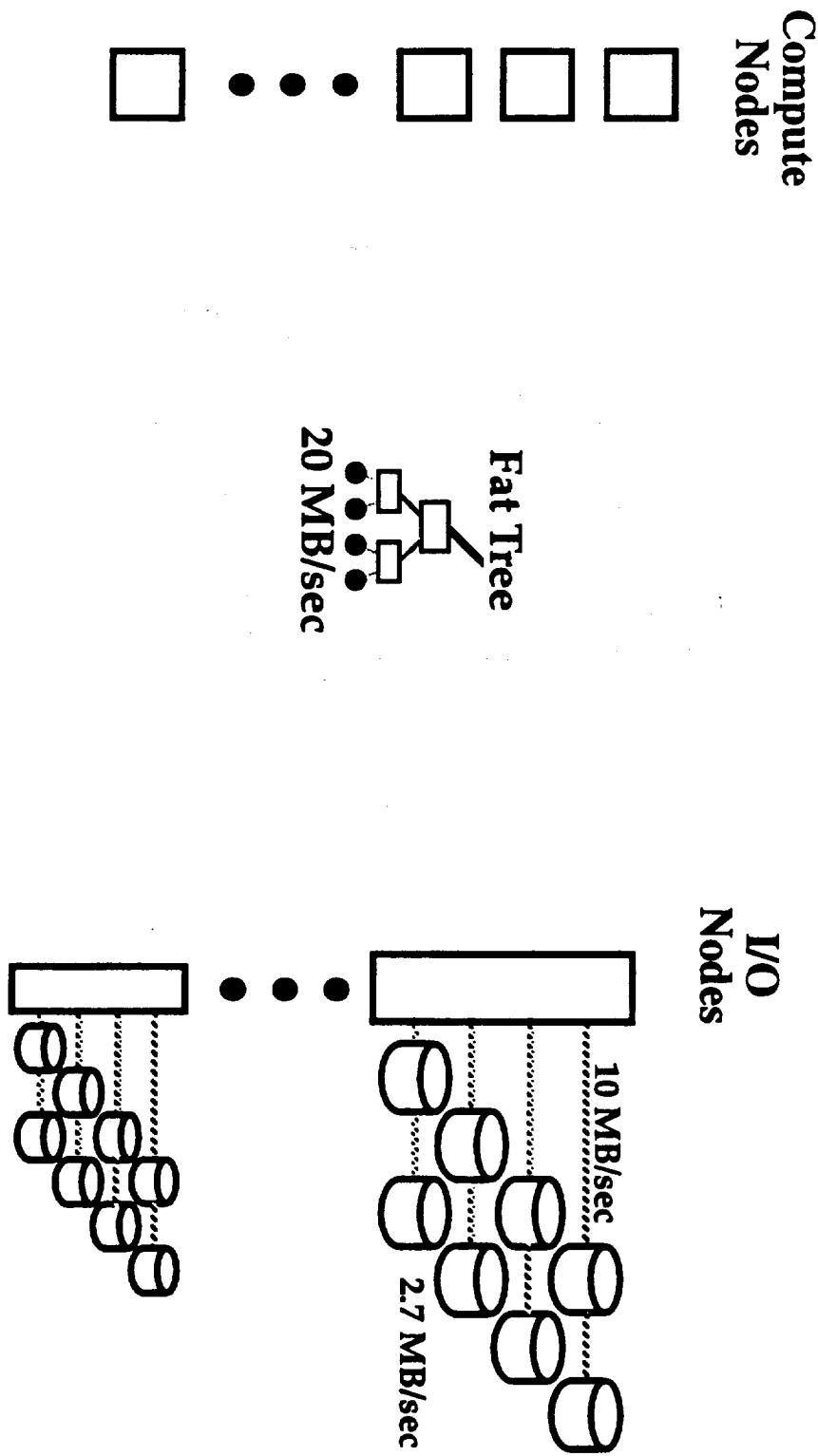
- Sustains 50-75% of peak disk rate
- Requires special tactics for read performance
- Scales with I/O and compute nodes (see above)

## Balance:

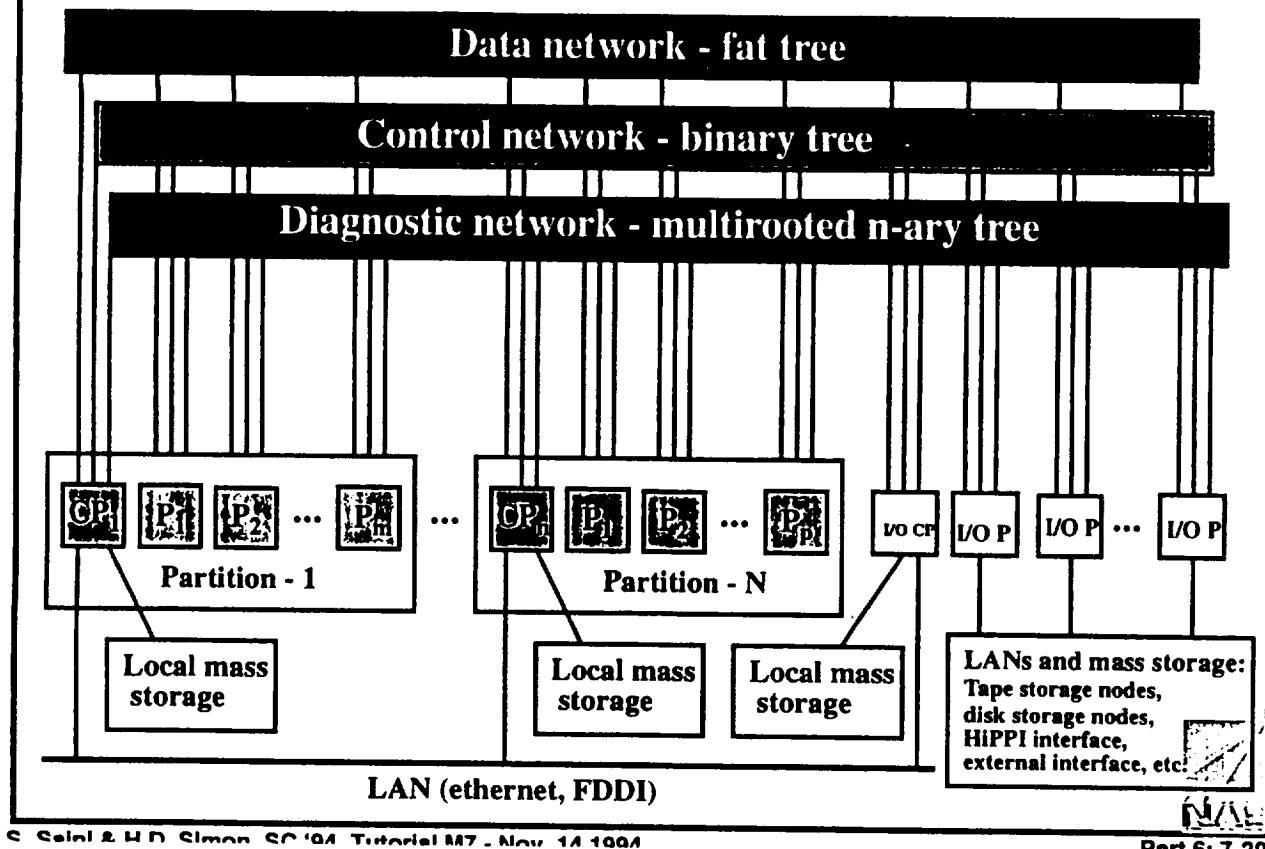
iPSC/860 (128 PE/10 IO)	Processor(s) MFLOPS	Network MB/sec	I/O Node(s) MB/sec
Single Node	60	2.8	1
System	7600	358	7.5** <sup>a</sup>

a. Assumes grouping

# TMC CM-5 I/O Architecture



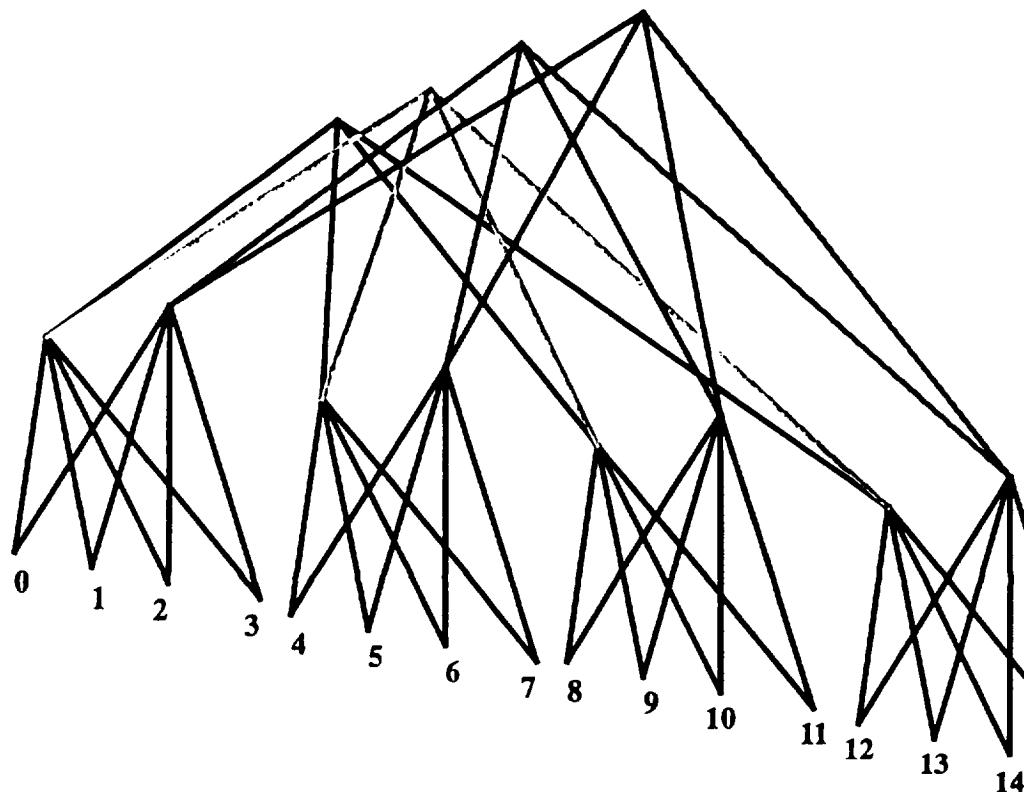
# TMC CM-5E Overall Architecture



© Seidel & H.D. Simon, SC '94 Tutorial M7, Nov. 14 1994

Part 6: 7.29

## Data Network with 16 Nodes



NASA  
SC '94

# TMC CM-5E System

Characteristic	Specification
Latency	80 microsecond
Bandwidth	Peak - 20 MB/s; measured 4-10 MB/s
Network topology: Data network Control network Diagnostic network	Three networks: 4-ary fat tree, 20 MB Binary tree (global operations) ; 20 MB/s Multirooted n-ary tree; 100 Mbit/s
Bisection bandwidth	940 MB for 128 nodes
Distributed DRAM memory	8 or 32 MB per node
Peak Mflop/s	16 Gflop/s for 128 nodes
Number of nodes	32 to 512 nodes
Operating System	CMOST
Languages	F77, CM Fortran , C*, LISP*, HPF*
Message passing library	PVM, CMMD (native library), and MPI
Scalable Disk Array (SDA)	48 GB

# **sfs: A Parallel File System for the CM-5**

**Susan LoVerso**

**Marshall Isman**

**Andy Nanopoulos**

**Bill Nesheim**

**Ewan Milne**

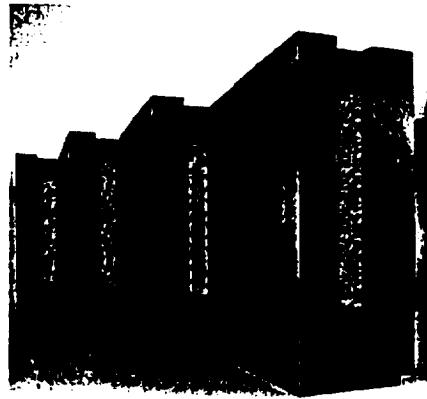
**Rick Wheeler**

**Thinking Machines Corporation**

**Cambridge, MA**

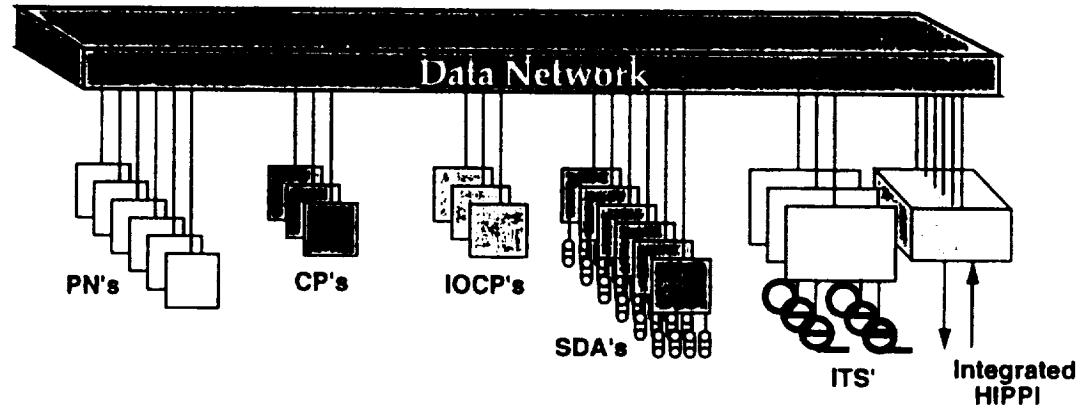
**Thinking Machines Corporation**

## **The CM-5**



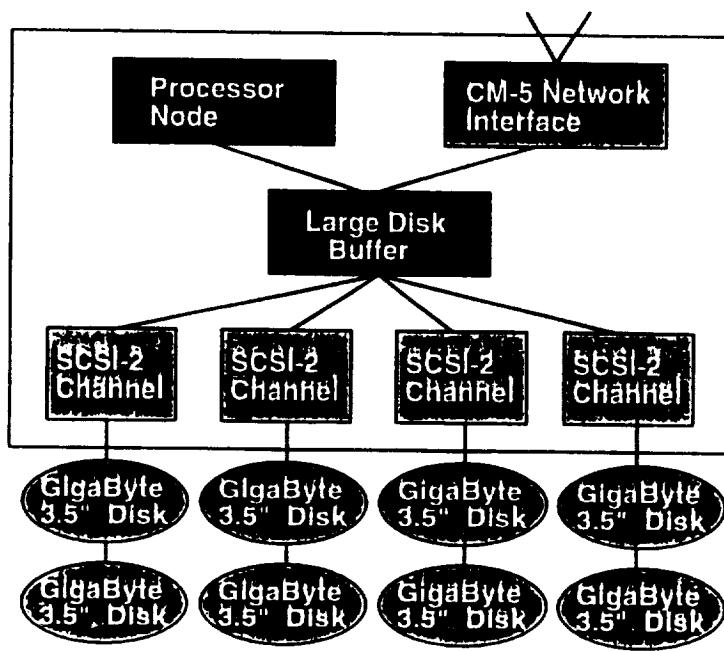
**Thinking Machines Corporation**

# CM-5 Hardware Overview



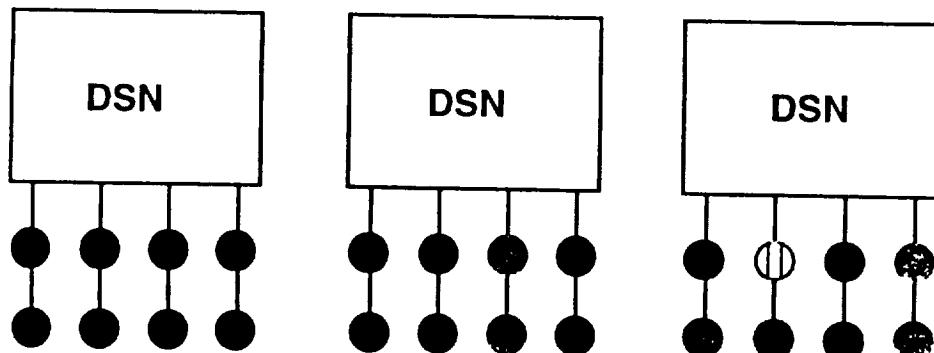
Thinking Machines Corporation

## SDA Hardware (DSN)



Thinking Machines Corporation

# Logical Devices



- 7 data disks + 1 parity disk -
- 6 data disks + 1 parity disk -
- 8 data disks + 0 parity disks -
- 1 spare disk

Thinking Machines Corporation

## RAID - 3

- Striping < 1 block per disk
- One designated parity disk
- Pre-disposed to large I/O

Definitions: - disk stripe: # data disks \* amt per disk - logical block: # data disks \* sector

Thinking Machines Corporation

## **sfs Goals**

- Provide an interface for parallel I/O calls
- Provide Unix compatibility
- Support very large files
- Support strange block sizes
- Create large contiguous files

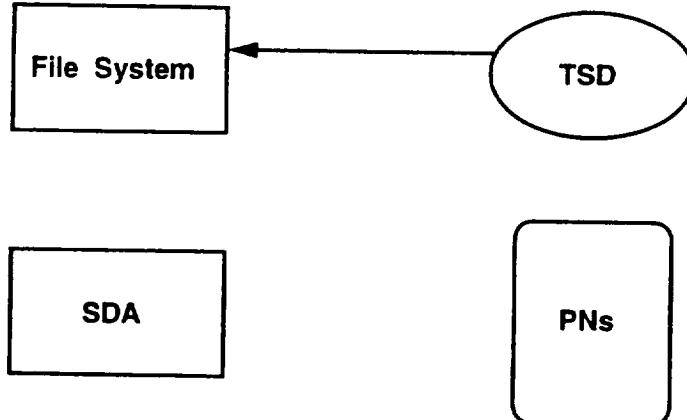
Thinking Machines Corporation

## **Parallel I/O**

- TSD interfaces between nodes and file system
- TSD acts as device driver for SDA
- File system provides TSD with block information

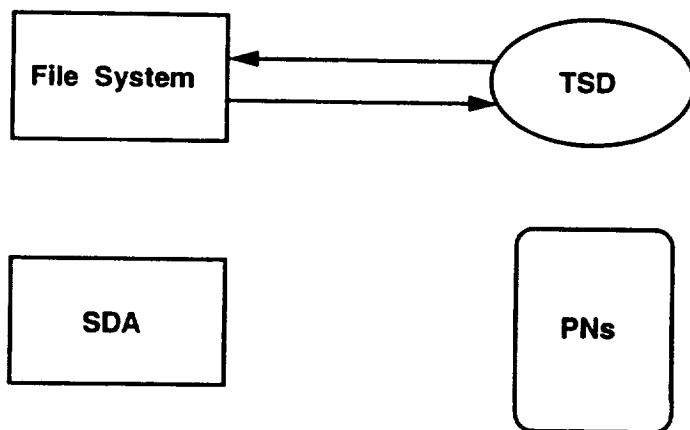
Thinking Machines Corporation

## Start of Parallel I/O



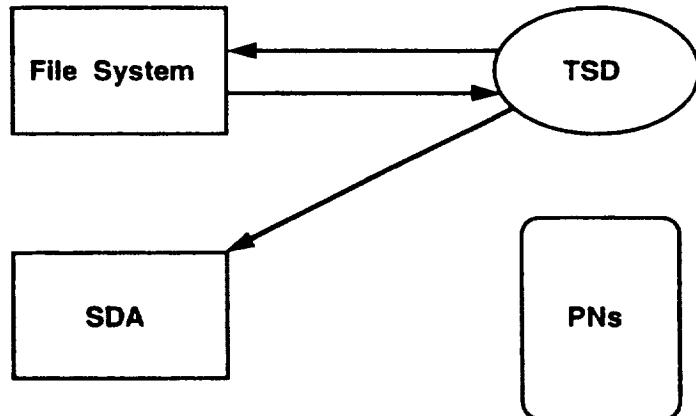
Thinking Machines Corporation

## Start of Parallel I/O



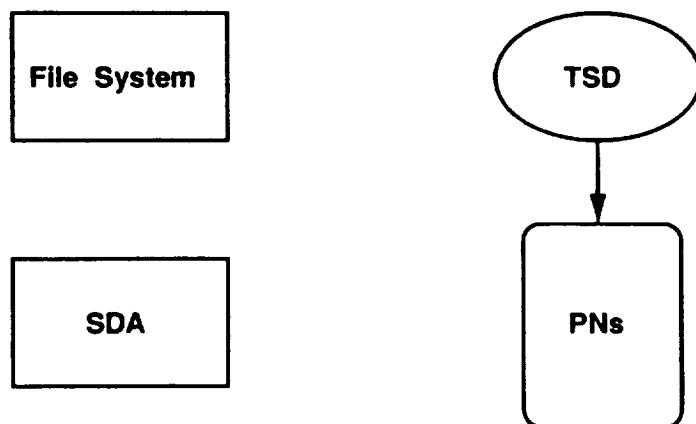
Thinking Machines Corporation

## Start of Parallel I/O



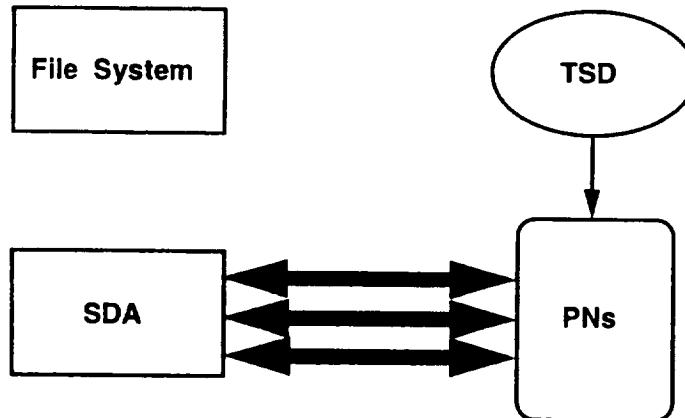
Thinking Machines Corporation

## Transferring the Data



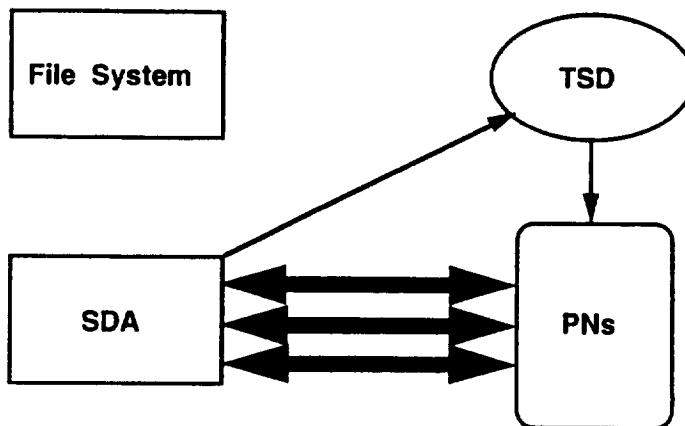
Thinking Machines Corporation

## Transferring the Data



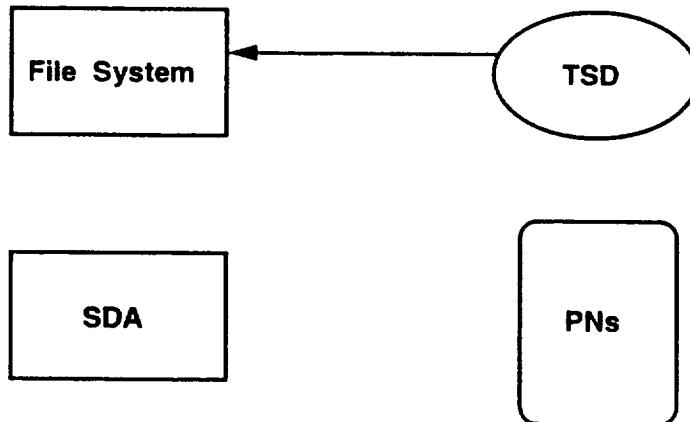
Thinking Machines Corporation

## Transferring the Data



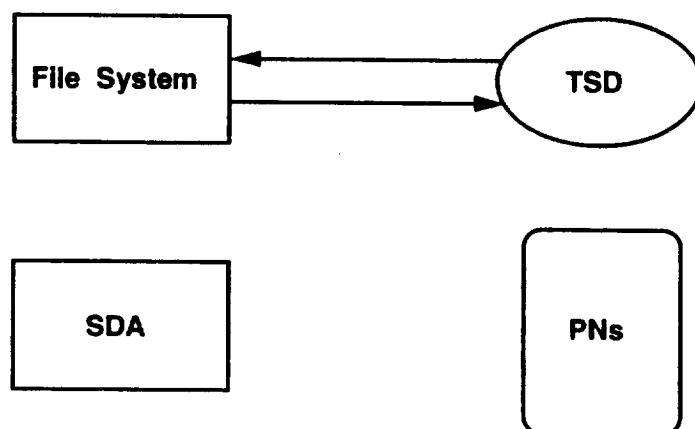
Thinking Machines Corporation

## Finishing the I/O



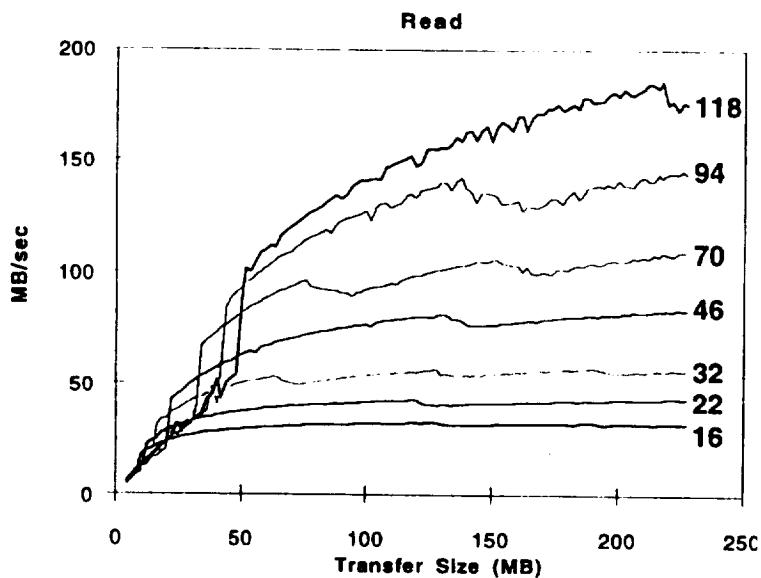
Thinking Machines Corporation

## Finishing the I/O



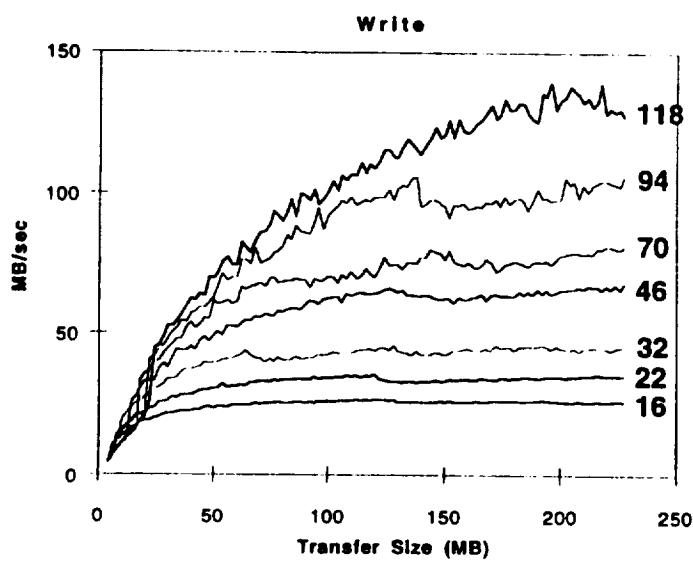
Thinking Machines Corporation

## Parallel Read Performance



Thinking Machines Corporation

## Parallel Write Performance



Thinking Machines Corporation

# CM-5 Summary

## File System: SDA

- Pseudo-SIMD, modified UNIX file system
- Files striped in 16 byte units across disks
- Software RAID-3

## I/O nodes:

- Vector-less compute nodes — SPARC w/16 MB
- Attached inside fat tree (20 MB/sec)
- 8 SCSI-2 IBM 0663E15 1.2 GB disks (14.4 MB/sec)

# CM-5 Summary, cont.

## Performance

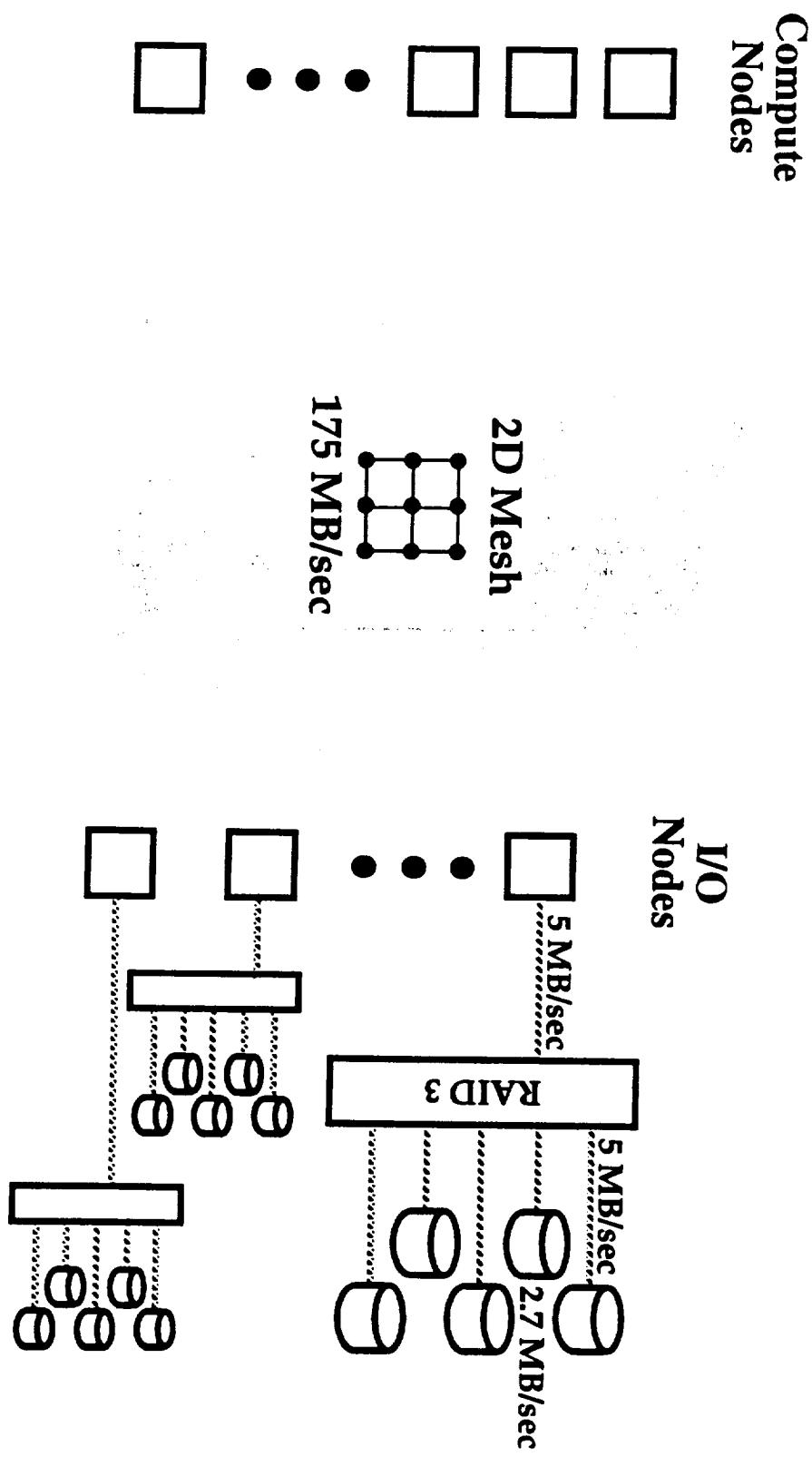
- Sustains 55-75% of peak disk rate
- Scales well with I/O and compute nodes

## Balance:

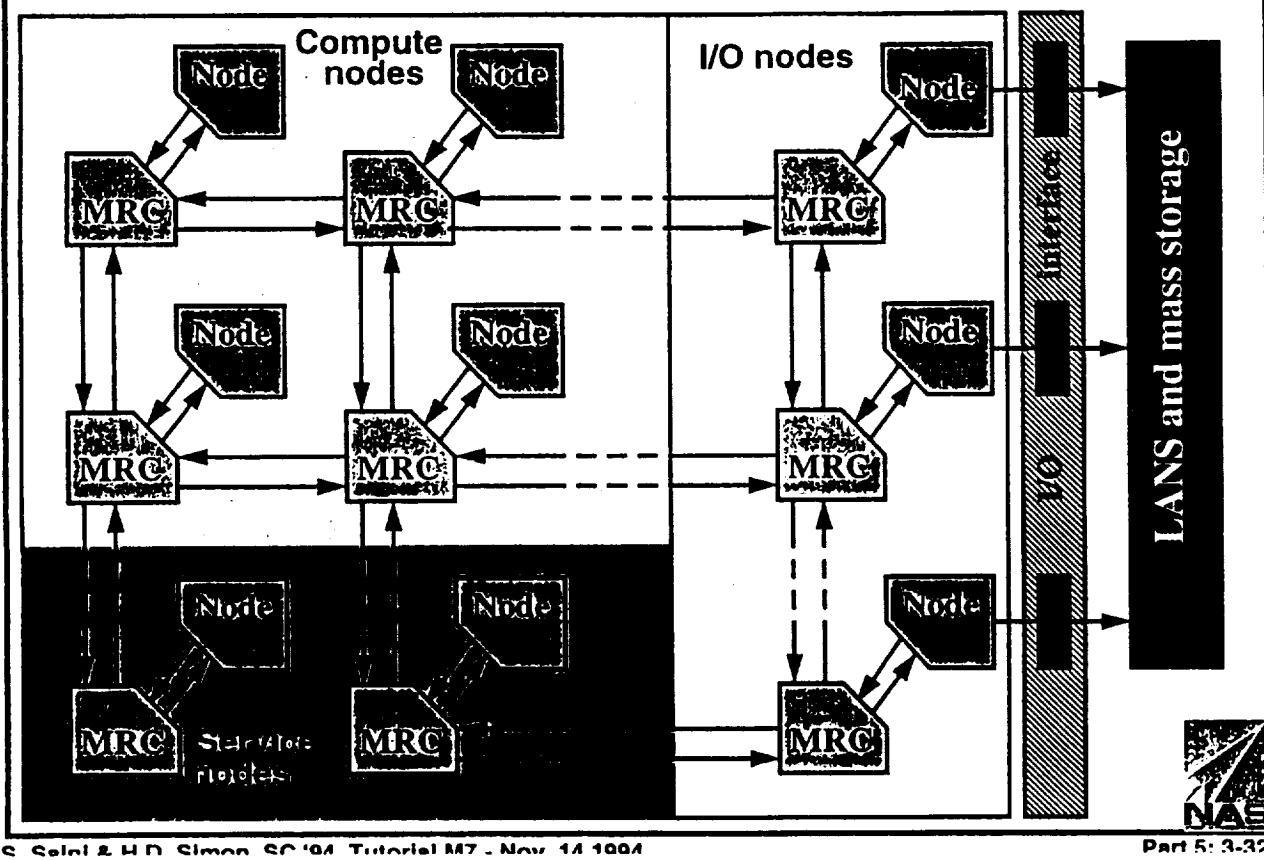
CM-5 (128 PE/6 IO)	Processor(s) MFLOPS	Network MB/sec	I/O Node(s) MB/sec
Single Node	160	20	14.4
System	20480	940	66.* <sup>a</sup>

a. Write performance for 46 disks

# Intel Paragon I/O Architecture



# Intel Paragon Overall Architecture



© Scott & H D Simon SC '94 Tutorial M7 - Nov 14 1994

Part 5: 3-32

## Intel Paragon System

Characteristic	Specification
Latency	80/45 (R1.1/R1.2) micro second
Bandwidth - bidirectional	Peak - 175 MB/s; measured 35/90 MB/s (OSF)
Topology	Wormhole routing packet
Bisection bandwidth	5.6 GB per second (256 nodes)
Distributed DRAM memory	16 - 32 MB per node
Peak Mflop/s	15.2 Gflop/s for 204 compute nodes
Number of nodes	16 to 1890 nodes
Operating System	OSF /1AD and SUNMOS
Languages	F77, F90, C, C++, HPF*
Message passing library	PVM, NX, and MPI*
Network Bandwidth	175 MB per second
Ethernet ports	1 per node



# **Paragon Parallel File System**

**Brad Rullman**

Brad Rullman

Intel Supercomputer Systems Division

6/24/94

1

## **Paragon PFS Objectives**

- Scalable I/O performance
- Large transfers
- Large files
- Flexible and extensible
- Compatible with standard OSF/1 system calls and commands
- Source-compatible with iPSC/860 CFS

Brad Rullman

Intel Supercomputer Systems Division

6/24/94

# PFS User Model

- Implemented as an OSF/1 mountable file system
- All “regular” files in the file system are striped
- Multiple PFS file systems may be mounted with different default stripe attributes
- Mounted anywhere in system-wide namespace
- Standard OSF/1 commands work
  - E.g. ‘cp’ and ‘mv’ work between UFS and PFS file systems

Brad Rullman

Intel Supercomputer Systems Division

6/24/94

5

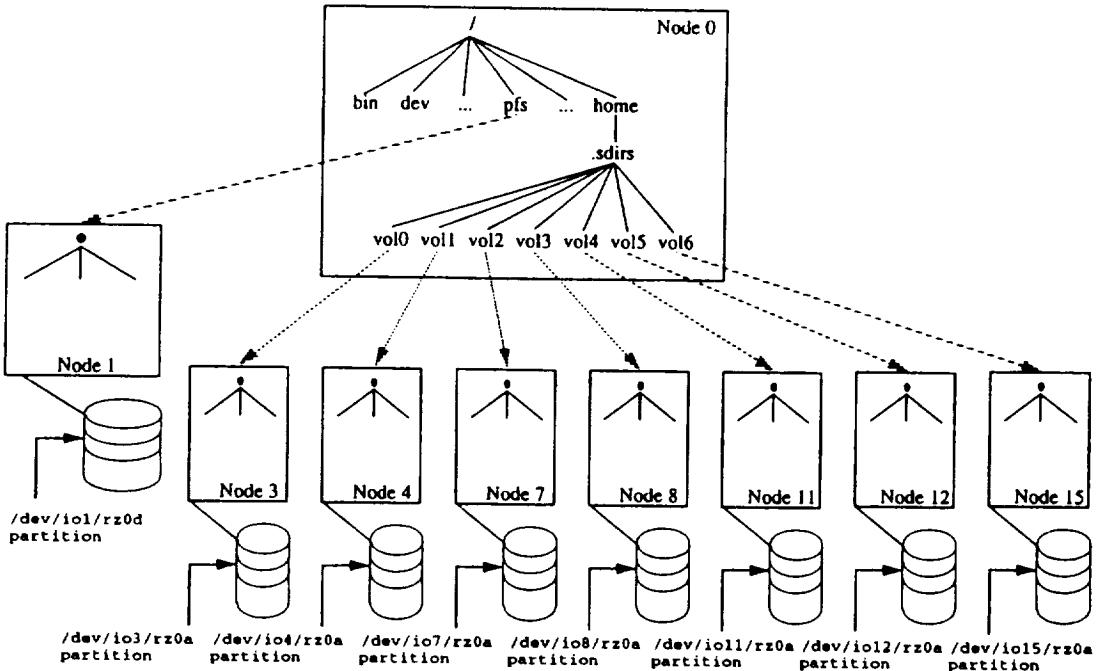
## Stripe Attributes

- Currently consist of:
  - stripe unit size
  - stripe factor
  - stripe group (stripe data storage locations)
- Stored as per-file meta data
- Cached in the compute node
- Extensible

Brad Rullman

Intel Supercomputer Systems Division

# Mounting a PFS



Brad Rullman

Intel Supercomputer Systems Division

6/24/94

7

## NORMA Limitations Affecting PFS

- **R1.2 Bandwidth limitation (single compute node)**
  - 17 MB/sec outgoing, 10 MB/sec incoming
- **“Many-to-one” limitation**
  - Avoid greater than 32:1 compute:I/O node ratio
  - Use `gopen()` rather than `open()` if possible, to avoid many simultaneous open requests for the same file
- **Very large aggregate I/O requests**
  - Success depends on physical memory available on I/O node
  - With 16 MB I/O nodes, keep simultaneous aggregate reads or writes to any one I/O node <= 10 MB
  - With 32 MB I/O nodes, keep simultaneous aggregate reads or writes to any one I/O node <= 26 MB

Brad Rullman

Intel Supercomputer Systems Division

SPARC

# Request Buffers

- Align I/O buffers on an 8 KB boundary (virtual memory page size)
  - Improves Mach IPC performance
- Read or write multiples of whole file system blocks (typically 64 KB for PFS)
  - Improves file system performance

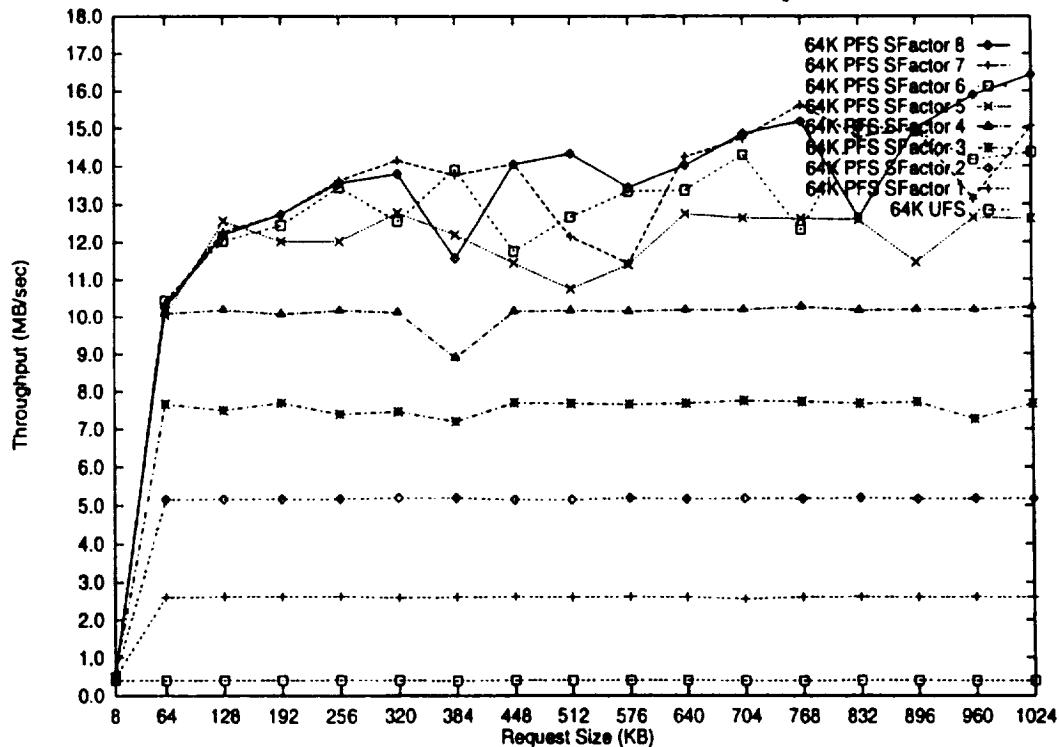
Brad Rullman

Intel Supercomputer Systems Division

6/24/94

14

## PFS Write Performance from One Compute Node

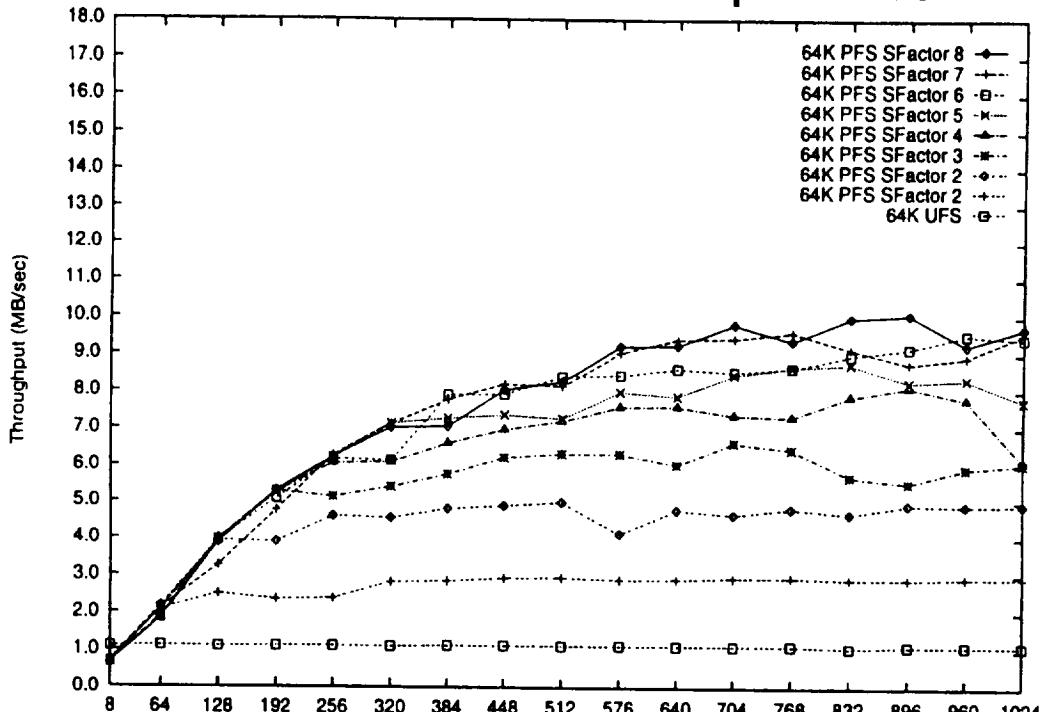


Brad Rullman

Intel Supercomputer Systems Division

6/24/94

## PFS Read Performance to One Compute Node



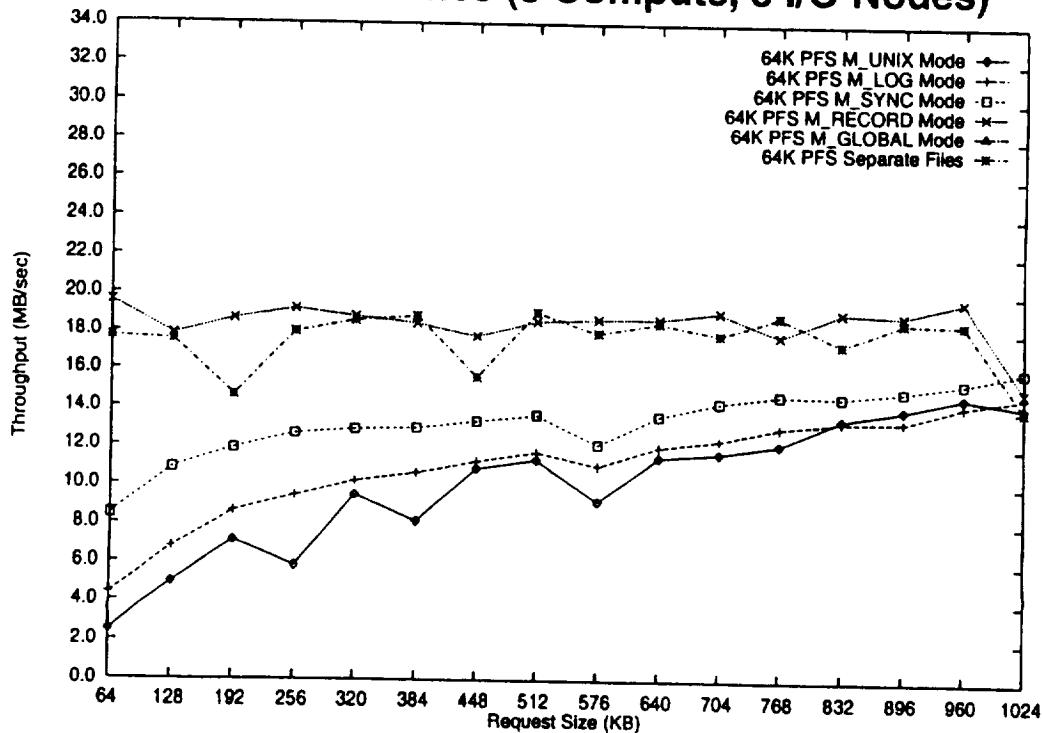
Brad Rullman

Intel Supercomputer Systems Division

6/24/94

17

## PFS Write Performance (8 Compute, 8 I/O Nodes)

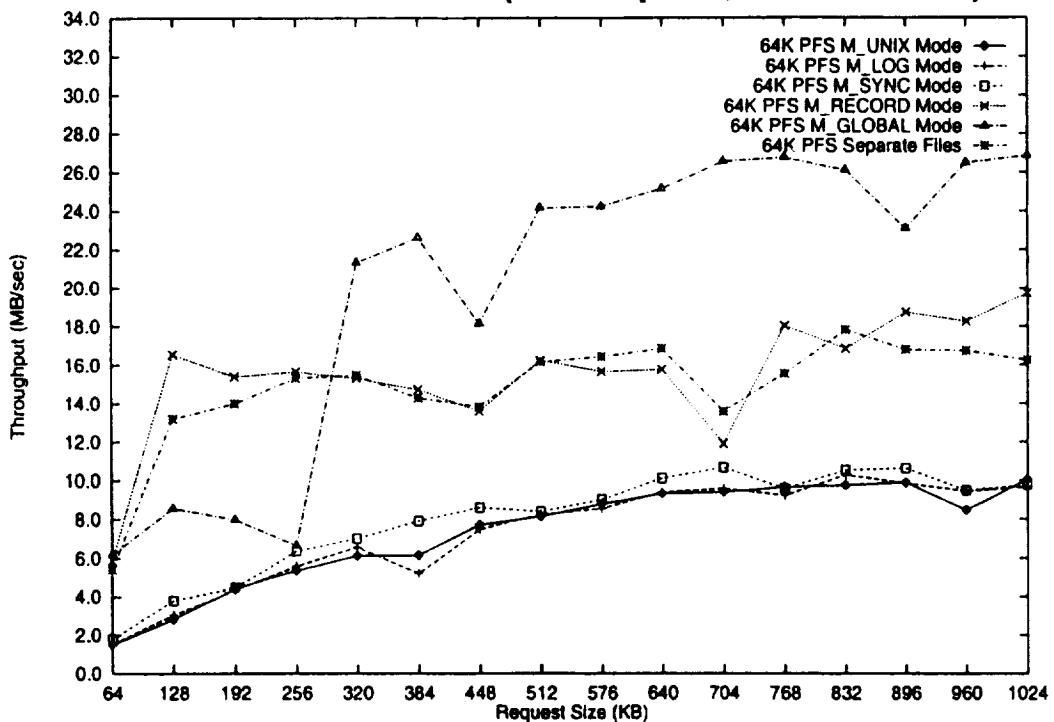


Brad Rullman

Intel Supercomputer Systems Division

6/24/94

## PFS Read Performance (8 Compute, 8 I/O Nodes)



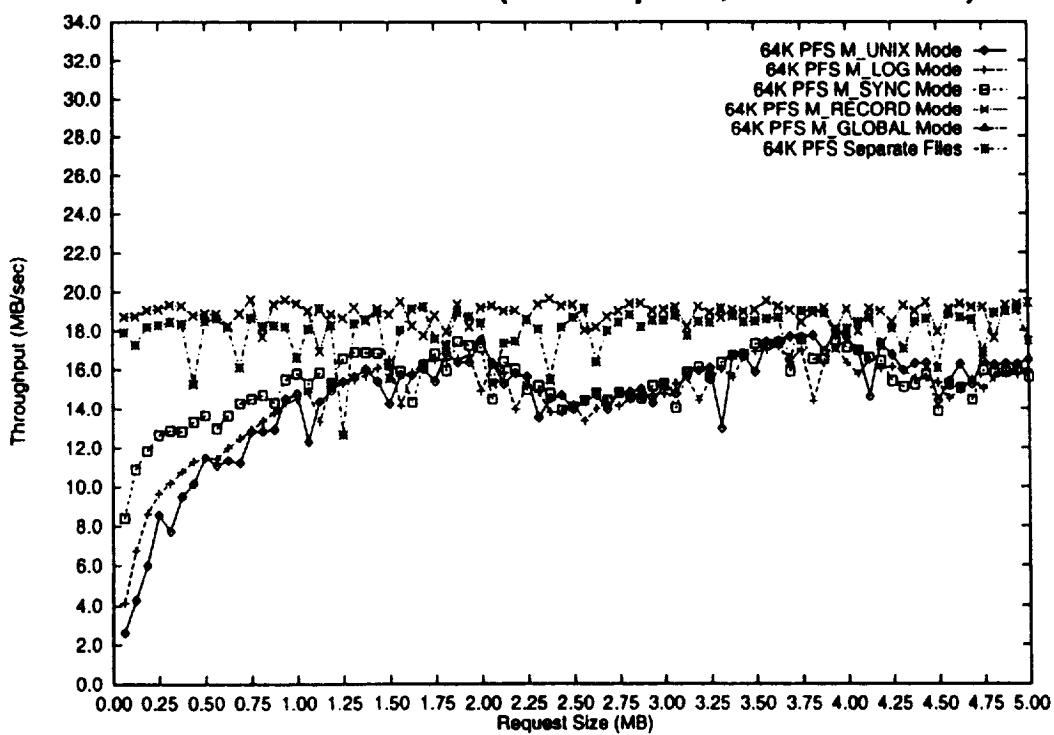
Brad Rullman

Intel Supercomputer Systems Division

6/24/94

19

## PFS Write Performance (8 Compute, 8 I/O Nodes)

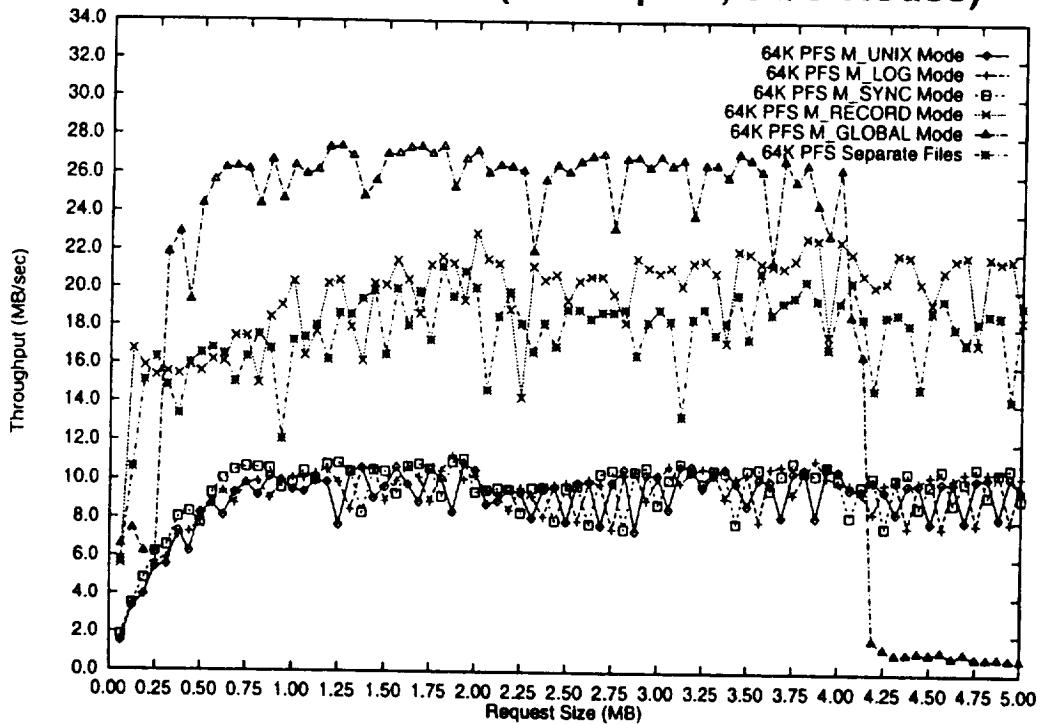


Brad Rullman

Intel Supercomputer Systems Division

6/24/94

## PFS Read Performance (8 Compute, 8 I/O Nodes)



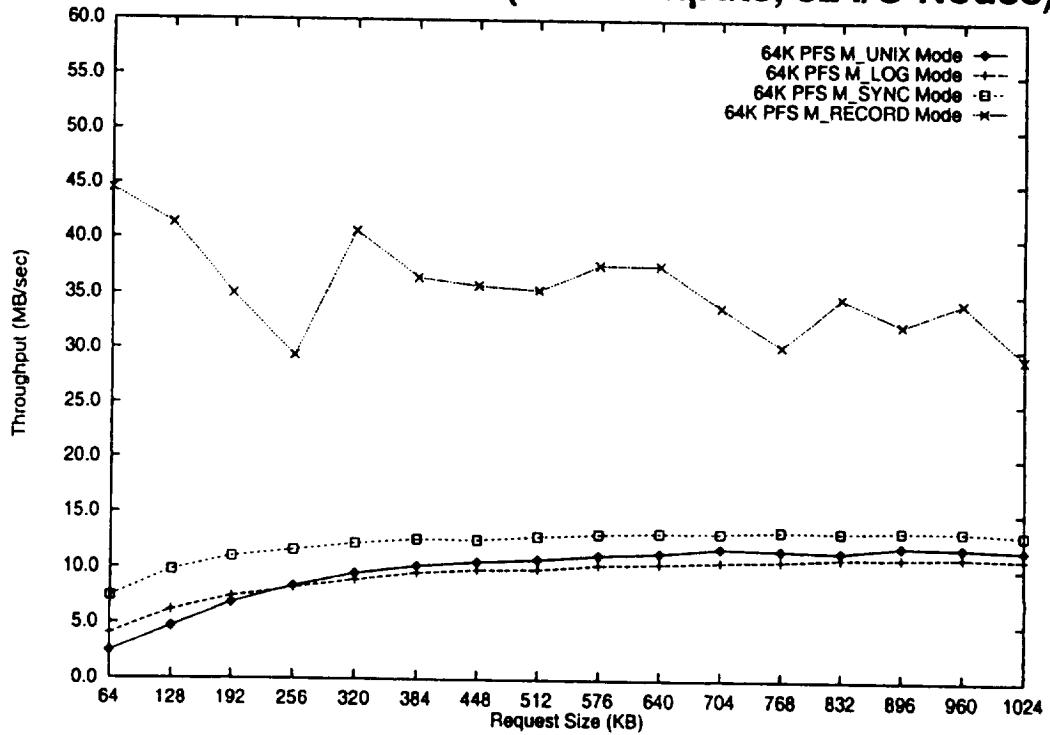
Brad Rullman

Intel Supercomputer Systems Division

6/24/94

21

## PFS Write Performance (256 Compute, 32 I/O Nodes)

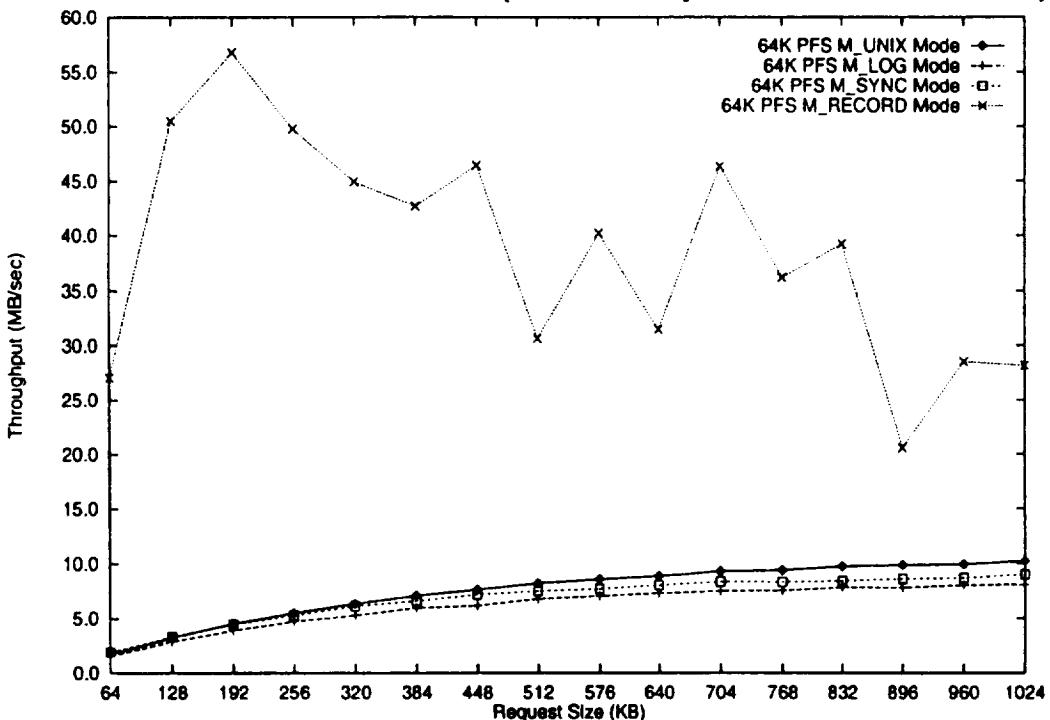


Brad Rullman

Intel Supercomputer Systems Division

6/24/94

## PFS Read Performance (256 Compute, 32 I/O Nodes)



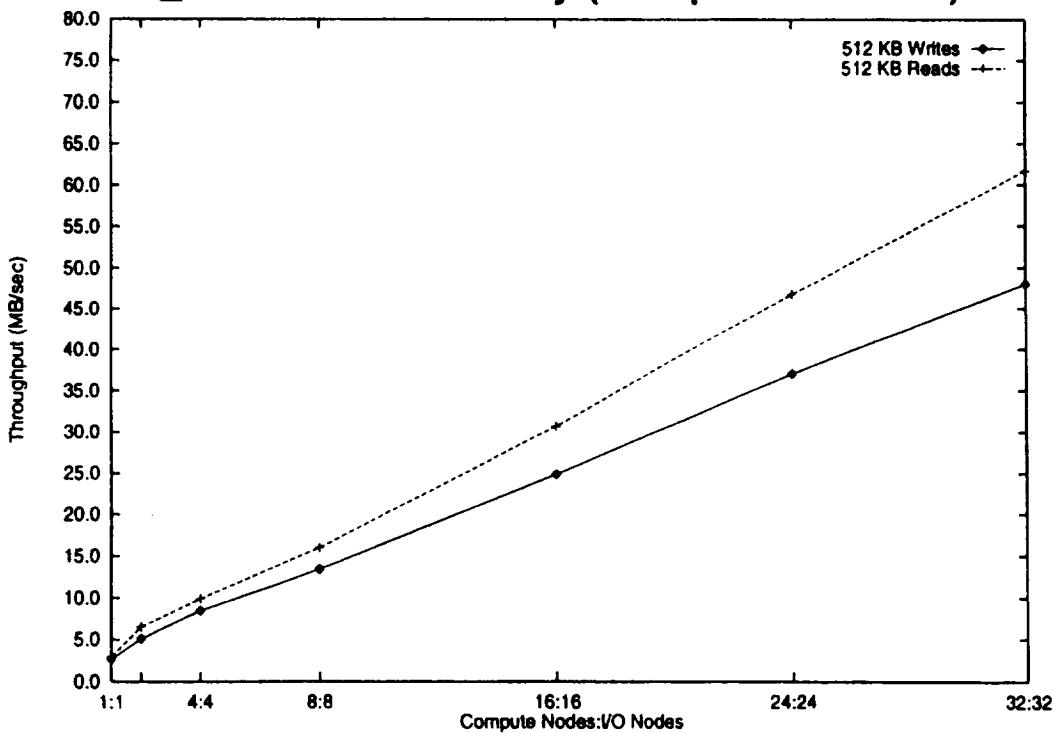
Brad Rullman

Intel Supercomputer Systems Division

6/24/94

23

## PFS M\_RECORD Scalability (Compute:I/O = 1:1)

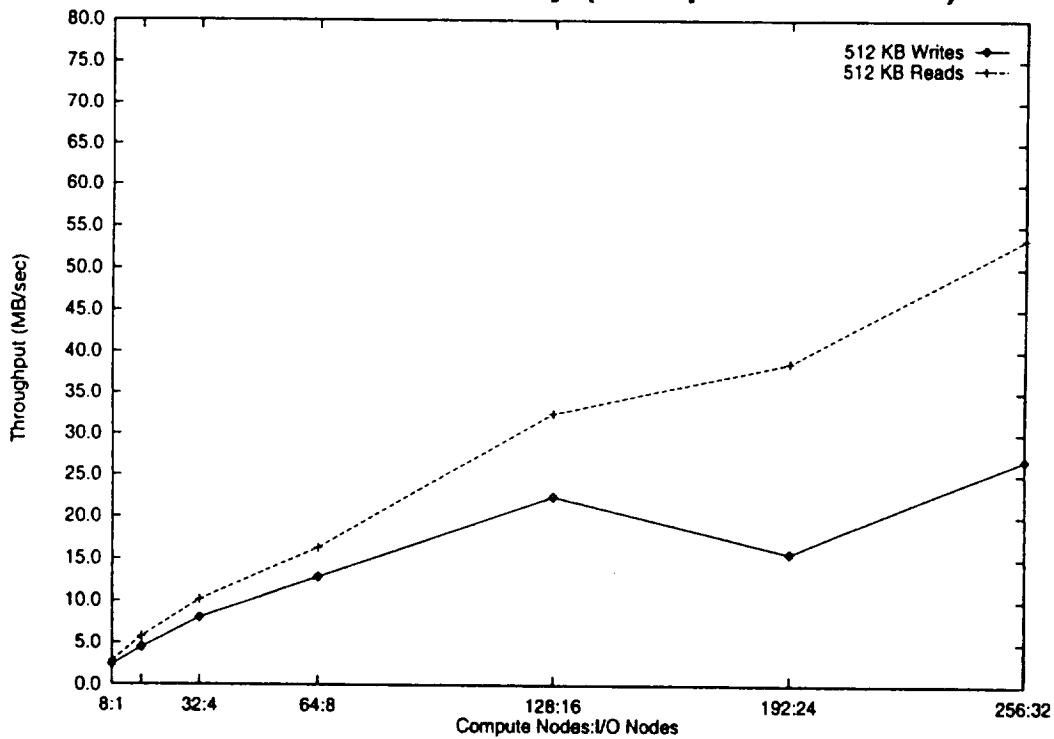


Brad Rullman

Intel Supercomputer Systems Division

SPARC

## PFS M\_RECORD Scalability (Compute:I/O = 8:1)

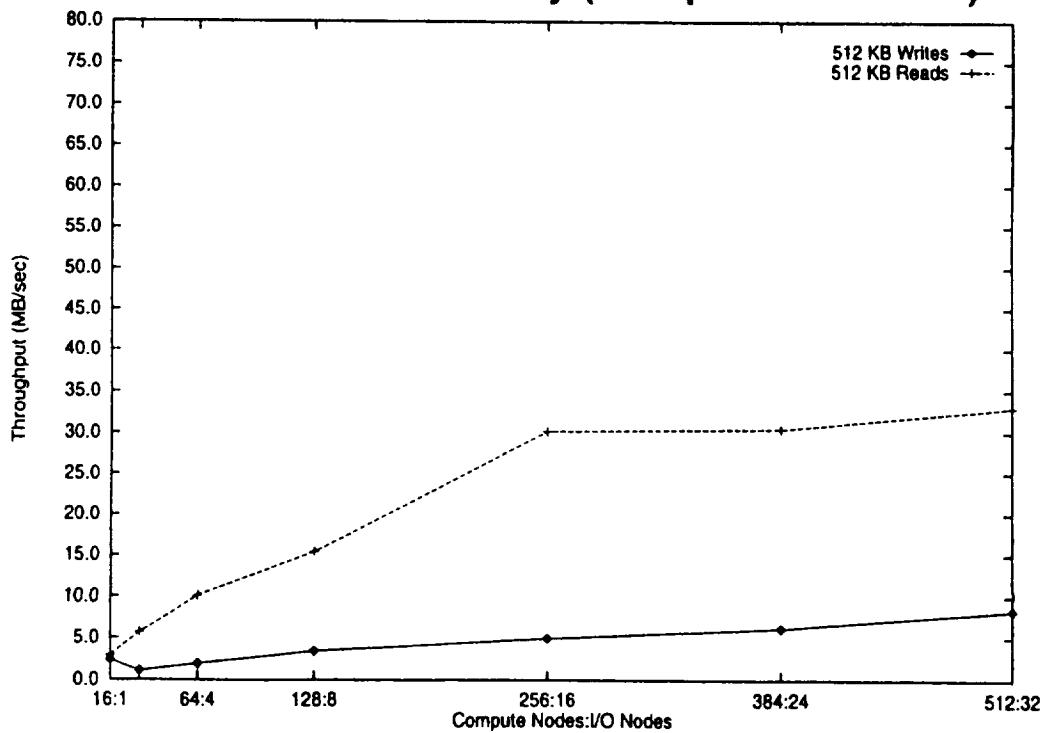


Brad Rullman ————— Intel Supercomputer Systems Division

6/24/94

29

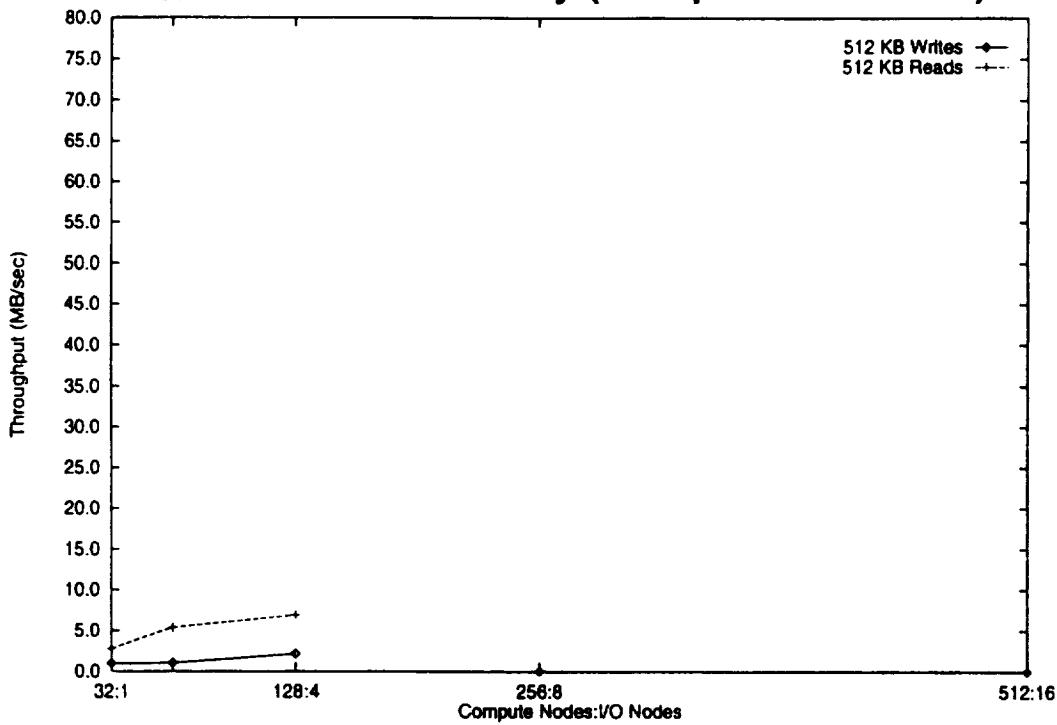
## PFS M\_RECORD Scalability (Compute:I/O = 16:1)



Brad Rullman ————— Intel Supercomputer Systems Division

6/24/94

## PFS M\_RECORD Scalability (Compute:I/O = 32:1)



Brad Rullman

Intel Supercomputer Systems Division

# Paragon Summary

## File System: PFS

- MIMD, UNIX file interface, built on top of UFS
- File blocks striped across UFS filesystems

## I/O nodes:

- Full compute nodes — Intel i860XP w/16-32 MB
- Attached inside mesh (175 MB/sec)
- 5 SCSI Maxtor 1240S 1.2 GB disks (2.5 MB/sec)
- Hardware RAID-3 (4+1)

# Paragon Summary, cont.

## Performance

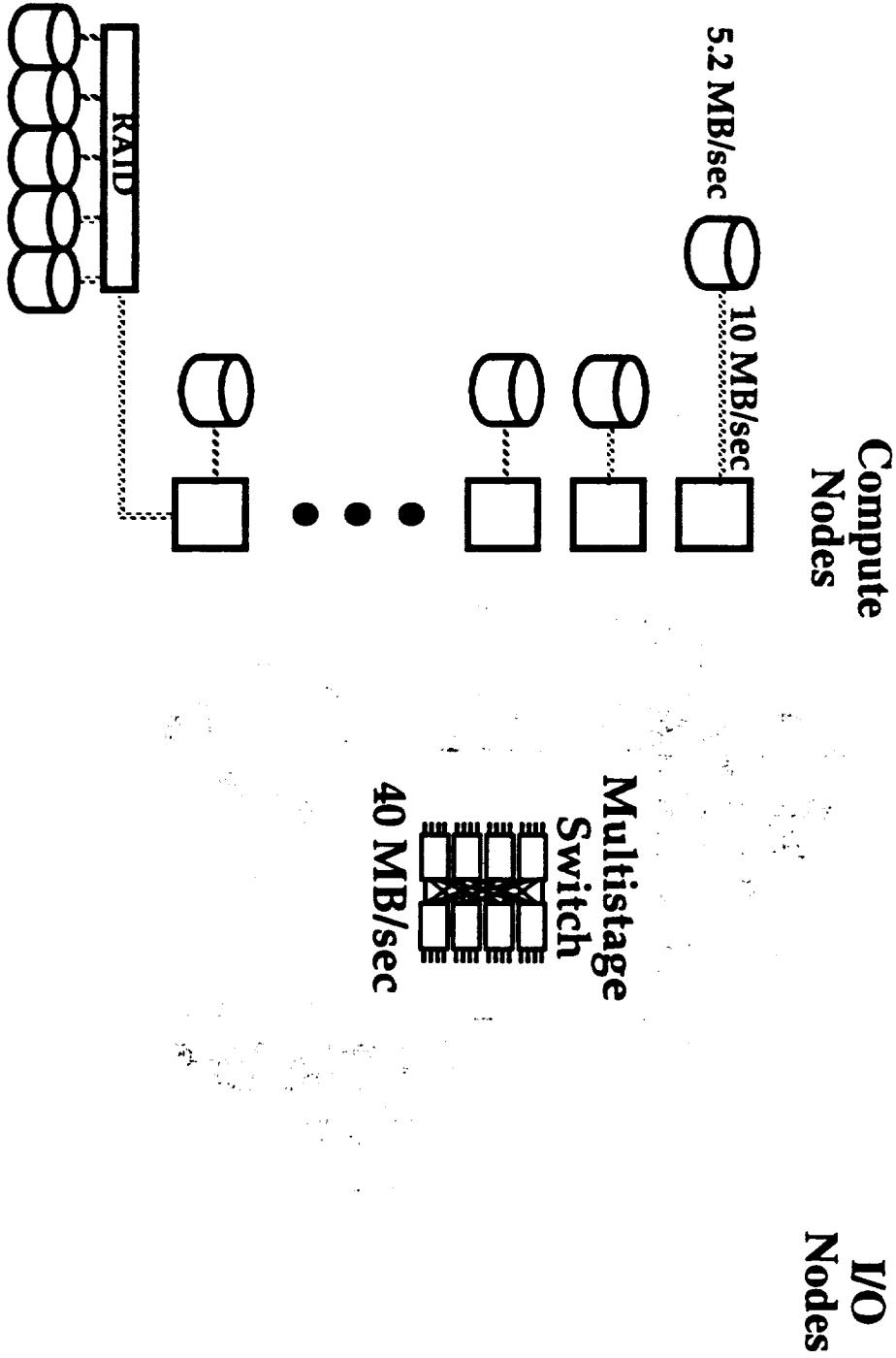
- Sustains 22-96% of peak I/O node rate
- NORMA (R1.2) interferes with good scaling

## Balance:

Paragon (208 PE/9 IO)	Processor(s) MFLOPS	Network MB/sec	I/O Node(s) MB/sec
Single Node	75	175	2.5
System	15600	5600	21.6** <sup>a</sup>

a. Assumes grouping; interpolated from 8 compute, 8 I/O nodes (-2.4 MB/sec per I/O node)

# IBM SP2 I/O Architecture



# IBM SP2 System

Characteristic	Specification
Latency	42 micro second
Bandwidth - bidirectional	Peak - 80 MB/s; measured 42 MB/s
Topology	Multistage, buffered wormhole routing packet-switch
Bandwidth - unidirectional	Peak 40 MB/s; measured 32 MB/sec.
Bisection bandwidth	10.2 GB per second
Distributed DRAM memory	32 -512 MB per node
Peak Mflop/s	68 Gflop/s for 256 wide nodes
Number of nodes	8 - 256 nodes. At NAS 128+32
Models available	"Wide" and "thin" nodes - 590 & 390 resp.
Operating System	AIX, IBM's Unix version
Languages	F77, F90, C, C++, HPF*
Message passing library	PVMe, MPL, and MPI*
Switch Bandwidth	40 MB per second
Ethernet ports	2 per node
External disk	24 GB - in addition to disks on each node



## SP2 Summary

File System: NFS/AFS — PFS "soon"...

- MIMD, UNIX workstation file system
- No support for >2 GB files, yet...

I/O nodes:

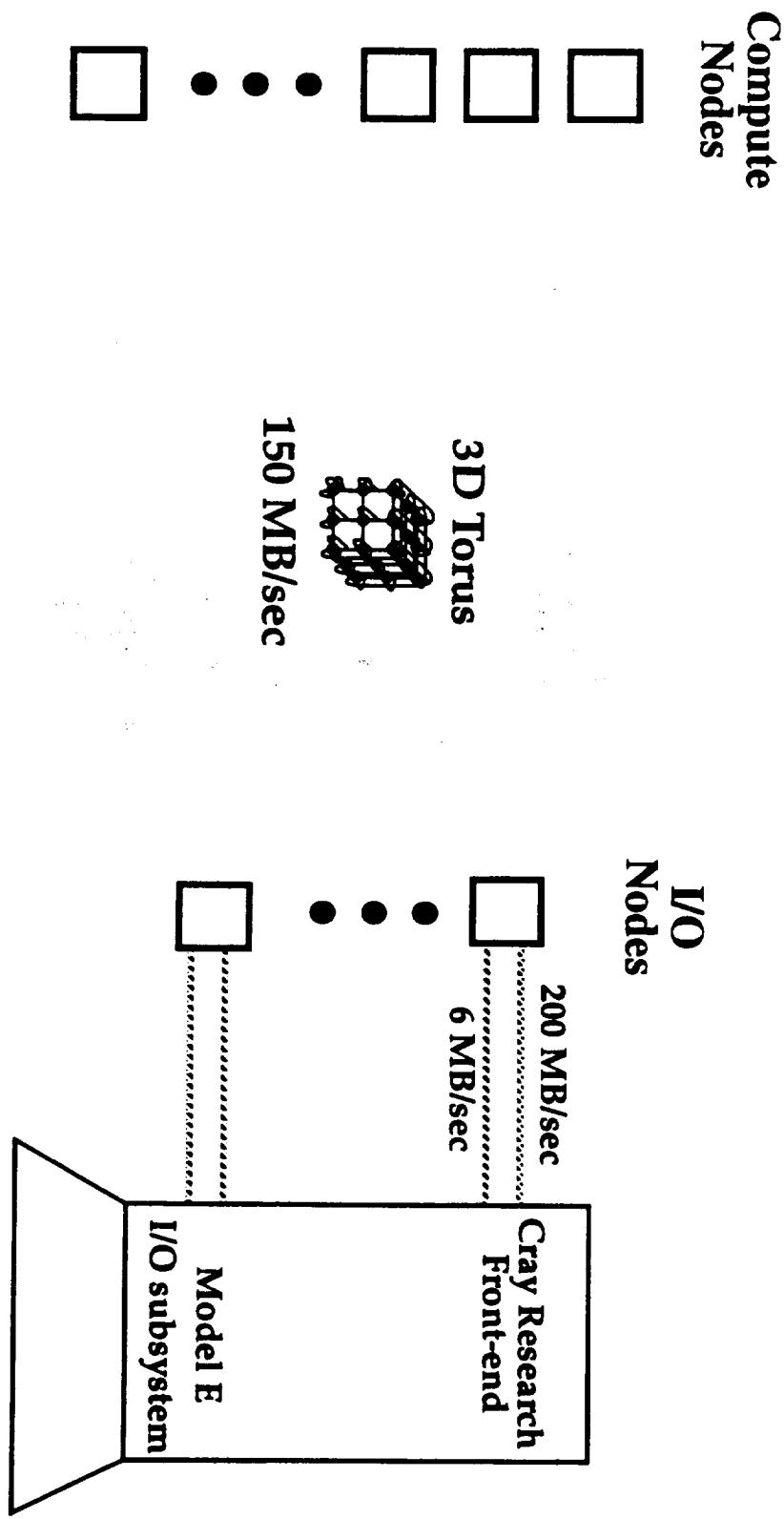
- Full compute nodes — every node has a disk
- Attached inside network (40 MB/sec)
- 1 SCSI-2 IBM 2580 2 GB disk (5.2 MB/sec)

Balance:

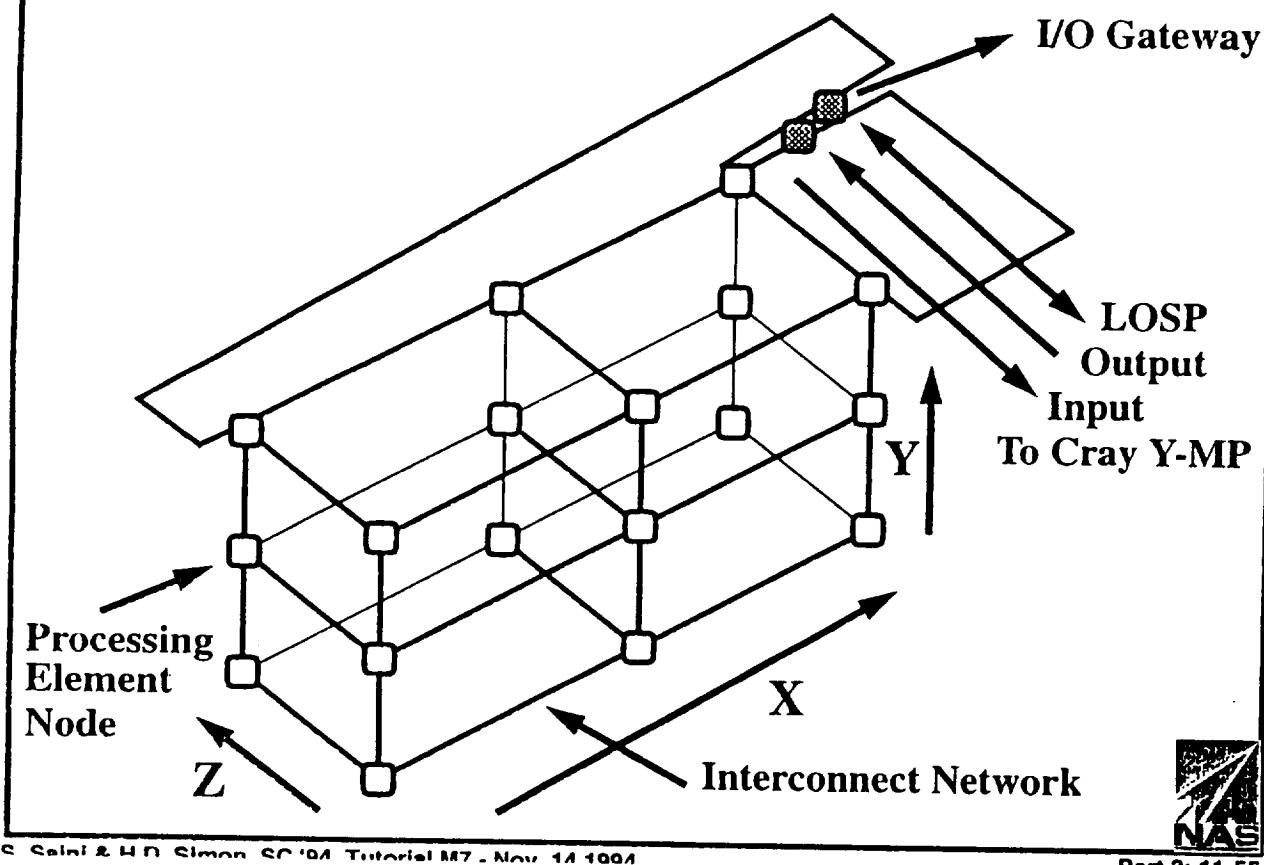
SP2 (160 PE/160 disks)	Processor(s) MFLOPS	Network MB/sec	I/O Node(s) MB/sec
Single Node	266	40	5.2
System	42560	10200	?



# Cray T3D I/O Architecture



# CRAY T3D System



C. Saini & H.D. Simon SC'04 Tutorial M7 - Nov. 14 2004

Part 3: 11-55



# CRAY T3D System

Characteristic	Specification
Number of microprocessors/node	2, one in each PE
Data transfer rate of LOSP channel	6 MB per second
Data transfer rate of HISI channel	200 MB per second
Topology	3-Dimensional torus - nodes interleaved
Bandwidth	Peak - 300 MB per second
Bisection bandwidth	76.8 GB per second for 1024 PEs
Distributed DRAM memory	16 MB/64 MB per PE
Peak MFLOPS	300 GFLOPS for 2048 PEs
Number of PEs	64 to 2048
Models available	Two, Single Cabinet and Multi Cabinet
Operating System	UNICOS - MAX
Microkernel	Resides on each PE (4 MB)
Message passing library	Based upon PVM with CRI extensions
Barriers/Eureka synchronization	



# T3D Summary

## File System: Cray host

- Centralized, UNIX file system on Cray host
- Cray host SSD can be used as file cache

## I/O nodes:

- Modified compute nodes — DEC Alpha w/16 MB
- Attached inside torus (X and Z only) (300 MB/sec)
- 1 HISP (200 MB/sec) connected to a Cray model E I/O system

# T3D Summary, cont.

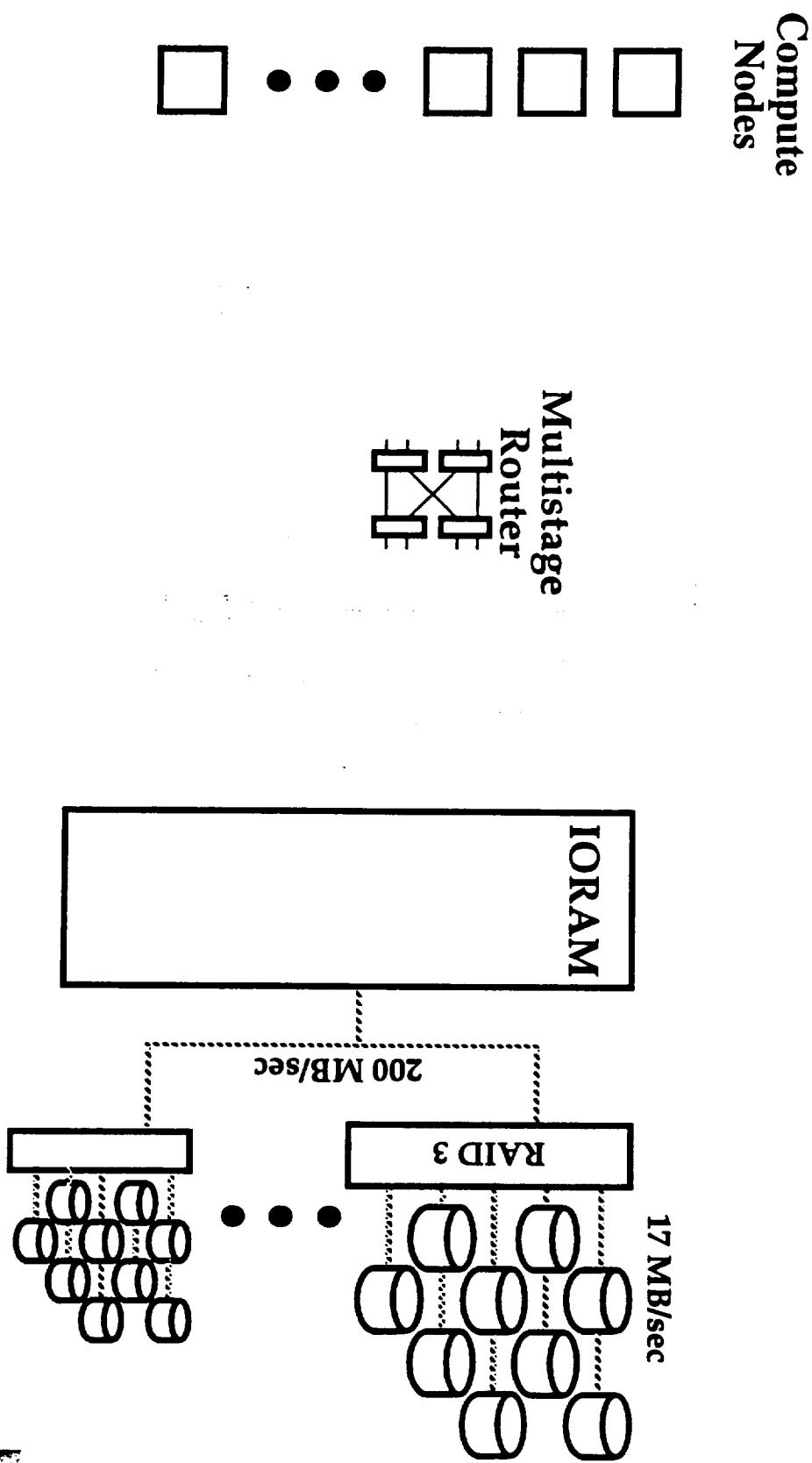
## Performance (with 4 I/O gateways)

- Writing to SSD: 570 MB/sec (71% of peak IOG rate)
- Reading from SSD: 400 MB/sec (50% of peak)
- Writing/reading physical disk: 360 MB/sec

## Balance:

T3D (256 PE/4 IOG)	Processor(s) MFLOPS	Network MB/sec	I/O Node(s) MB/sec
Single Node	150	300	200
System	38400	19200	360.*

# MasPar MP-2 I/O Architecture



# **The MasPar Scalable Unix I/O System**

---

*International Parallel Processing Symposium 1994*

**John R. Nickolls**

**MasPar Computer Corporation**  
**749 N. Mary Avenue**  
**Sunnyvale, CA 94022**  
**[nickolls@maspar.com](mailto:nickolls@maspar.com)**

---

**MASPAR**

**MasPar Computer Corporation**  
An IPPS™ MasPar I/O System

## **I/O System Design Goals**

---

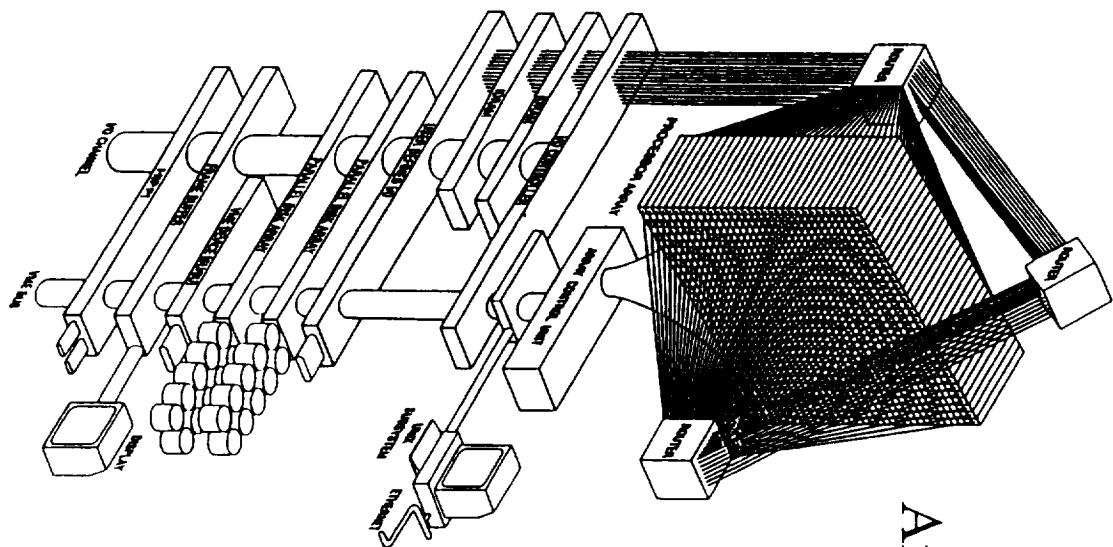
*Scalable I/O for distributed memory architecture*

- I/O performance balanced with compute performance
- I/O scalability with both processors and devices
- Programmable data mapping for distributed memory
- I/O independent of machine configuration
- Optimized for large files and large transfers
- Simple, cost-effective implementation

---

**MASPAR**

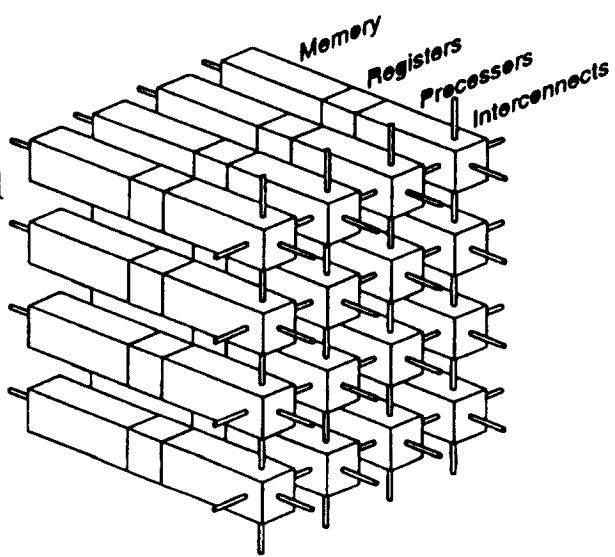
**MasPar Computer Corporation**  
An IPPS™ MasPar I/O System



## MasPar System Architecture

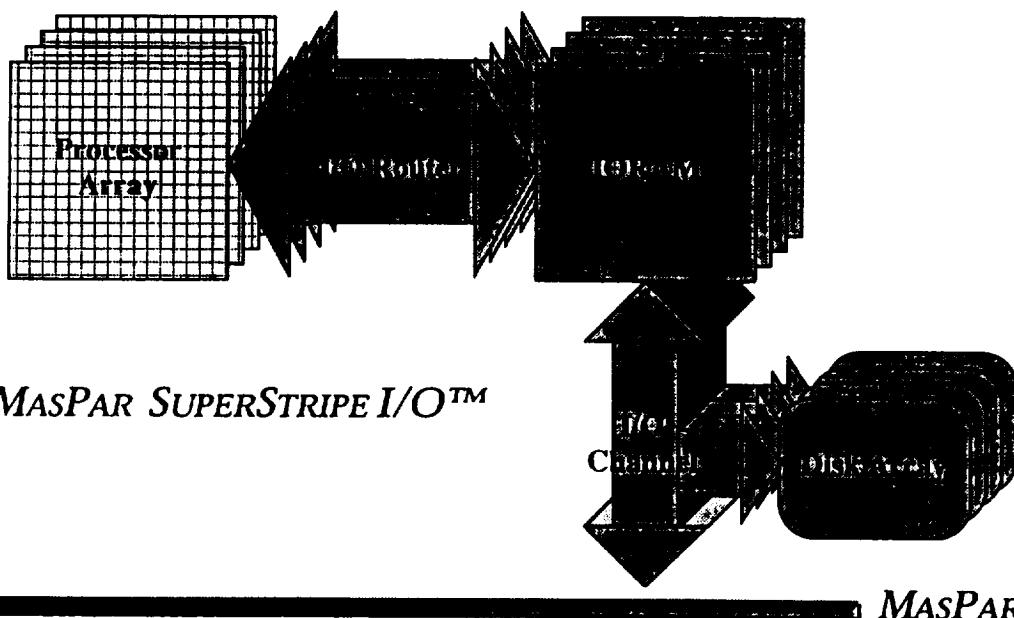
# Processor Array Architecture

- RISC Processor:
  - 40 32-bit Registers
  - Floating point 64, 32
  - Integer 64, 32, 16, 8, 1
- Processor Memory:
  - 16, 64, 256 KB
- Interconnects:
  - 2-D Torus Mesh
  - Multistage Router
  - Broadcast
  - Reduction



**MASPAR**

# MasPar I/O System Structure



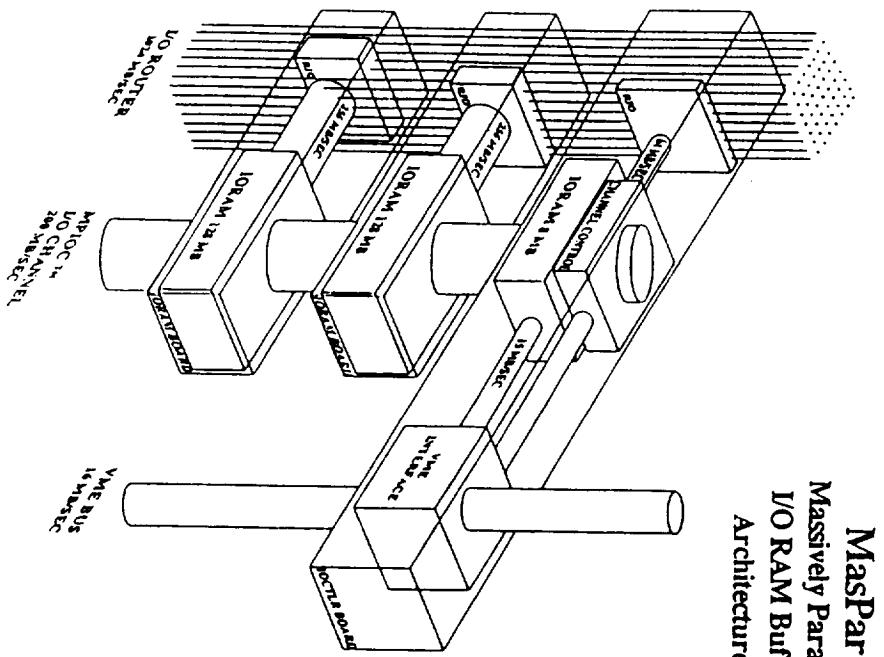
MasPar Computer Corporation  
MS-IPS™ MasPar I/O Systems

## IORM Buffer

*Interposed between router and I/O devices*

- Unix manages IORM as I/O buffer cache
- Conventional Unix byte stream I/O data organization
- IORM modules, up to 8 total
  - 32 MB or 128 MB buffer
  - 256 Router ports, 256 MB/sec
  - I/O Channel interface, 200 MB/sec
- Processors select target IORM position
  - Router connection with target RAM
  - RAM addressing for target byte
  - Bidirectional router connection

MasPar Computer Corporation  
MS-IPS™ MasPar I/O Systems



**MasPar**  
Massively Parallel  
I/O RAM Buffer  
Architecture

## MPDA: Scalable Disk Array

*RAID-3 plus RAID-0 Technology*

### ■ RAID-3 Disk Array Modules

- Data striping across disks with fault tolerance
- 11 GB capacity
- 18 MB/sec peak, 16 MB/sec sustained
- 8 data disks + 1 fault tolerance disk + optional hot standby

### ■ RAID-0 Arrays of Modules

- Data striping across RAID-3 modules
- Scalable performance up to 200 MB/sec
- Scalable capacity to 500 GB

**MASPAR**

# I/O System Software Goals

*Unix I/O for data-parallel applications*

- Data-parallel programming model
- Accommodate SPMD languages
- Simple extension of Unix programming model
- Unix file formats and data representations
- Parallel access to conventional Unix or Fortran files
- No “parallel file modes” for parallel access
- Intermixed compatible scalar and parallel I/O

**MASPAR**

MasPar Computer Corporation  
MPFS™ MasPar I/O System

# MPFS Unix File System

*Scalable file system block size*

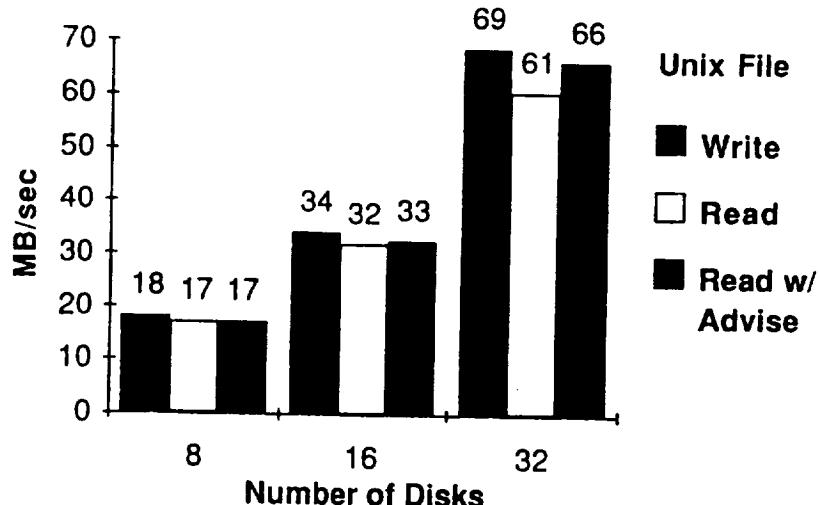
- Underlying block size scales with disk array performance
  - Constant ratio of overhead to transfer time
  - 16 KB per MB/sec
  - 256 KB block at 16 MB/sec
  - 1 MB block at 64 MB/sec
- Based on BSD Unix fast file system
  - File system buffers in IORAM
  - Extended buffer cache management
- NFS exportable
- File system performance scales with disk performance

**MASPAR**

MasPar Computer Corporation  
MPFS™ MasPar I/O System

# Scalable Disk Array Unix File Performance

*Parallel p\_read and p\_write of large files*

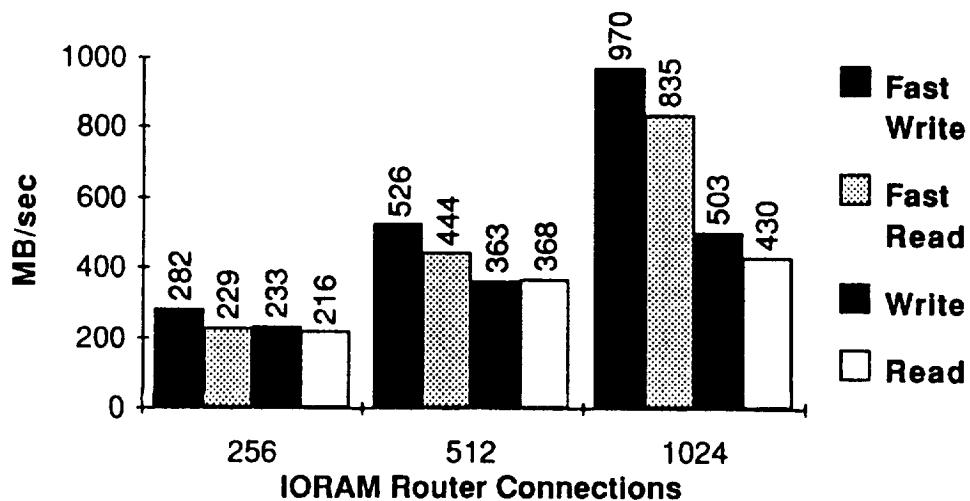


MasPar Computer Corporation  
MP-PPS 344 MasPar I/O System

*MASPAR*

# Solid-state RAM Disk Unix File Performance

*Parallel p\_read, p\_write, p\_fast\_read, p\_fast\_write*



MasPar Computer Corporation  
MP-PPS 344 MasPar I/O System

*MASPAR*

## MP-2 Summary

### File System:

- SIMD, UNIX UFS-based file system
- Use IORAM as filesystem cache—automatic data reorganization (e.g. transpose)

### No I/O nodes:

- IORAM attached via router network (1 MB/sec)
- Single I/O channel from IORAM to disk arrays (200 MB/sec)
- Disk array: 9 disks, 11 GB capacity (18 MB/sec)
- RAID-3 (8+1) per disk array

## MP-2 Summary, cont.

### Performance

- Sustains 88 - 96% of peak disk array rate
- Scales well with I/O and compute nodes

### Balance:

MP-1 (8K PE/4 arrays)	Processor(s) MFLOPS	Network MB/sec	I/O Node(s) MB/sec
Single Node			18
System	1200	640	69.*

# Architecture Summary

## Common Features

- UNIX file interface w/parallel extensions
- Files are striped across disks
- Support for > 2GB files (mostly...)
- Use of RAID-3 (software/hardware)
- Systems sustain over 50% of "peak"
- Good scaling (compute & I/O nodes), in theory...

## Interesting Differences

- MIMD, pseudo-SIMD, SIMD, centralized
- I/O node placement: inside/outside network
- Caching...and the need for special tactics

# System Balance

System (PEs/disks)	Processors MFLOPS	Network Bisection MB/sec	Single-file Write MB/sec	
			"Peak"	Sustained
C90 (16/80)	16000		740-1600	80-360
iPSC/860 (128/10)	7600	358	10	7.5*
CM-5 (128/48)	20480	940	86.4	66
Paragon (208/45)	15600	5600	22.5	21.6*
SP-2 (160/160)	42560	10200	832	?
T3D (256/4 IOGs)	38400	19200	800	360
MP-2 (8192/32)	1200	640	72	69

Performance = \$\$\$

# I/O Benchmarks

## What are Benchmarks For?

**Measuring how fast *your* applications will run**

**Determining the best I/O system *configuration***

**Determining the best way to *use* an I/O system**

- Collective/independent
- Synchronous/asynchronous

# How do "real" Applications Use I/O?

## Single Applications

## Workloads

- General workstation
- Scientific

# Measuring I/O Usage

## Ad-hoc analysis

- Examine representative applications
- Talk to scientists
- Emphasizes single applications

## Live workload measurement

- Trace generation added to library or OS
- Measures total system workload
- May be intrusive
- Need large sample

## Ad-hoc Analysis

### Example, time stepping flow solver

- Data written at periodic intervals for later visualization
- Data consists of 3D distributed arrays
- Block size dependant on data layout, can be very small for 3D decomposition
- Each write is generally 100s of MBytes (total across all nodes)

## CHARISMA Project

- Headed by David Kotz, Dartmouth College
- CHARacterize I/O in Scientific Multiprocessor Applications
- Measurements made on entire live workloads
- First work done by inserting trace generation into library on the NAS iPSC/860 (see talk by Kotz and Nieuwejaar in main conference)
- Future work planned on other systems



# iPSC/860 Workload

## Results (from Kotz and Nieuwejaar, SC '94 paper)

- Most jobs only open a few files
- Most files are large
- Most reads and writes are small
- Files either accessed 100% sequential or 100% non-sequential
- R/W files tended to be non-sequential, RO/WO tended to be sequential
- 70% of RO files had all bytes shared, 90% of WO files had no bytes shared

# Measuring Usage (summary)

## Examining applications

- Provides better information about why usage patterns exist
- Can slant evaluation towards a single type of application

## Workload evaluation

- Provides statistical information about entire workload
- Doesn't tell why patterns exist
- Need large sample size

# Measuring I/O Performance

## Serial (workstation) benchmarks

- Targeted towards workstation problems
- e.g., many seeks, small files, files not shared

## Scientific benchmarks

- Emulate scientific application requirements
- NAS Parallel I/O Benchmarks

## The NAS Parallel I/O Benchmarks

Tests three main categories of I/O used at NAS

- System (*peak*) disk
- ~~Network~~
- Application disk

Designed to test both aggregate system performance and balance

- Fast I/O or compute performance alone is insufficient
- Extra I/O or compute performance may not affect aggregate performance if a system is out of balance

# Peak Disk I/O Benchmark

## Write test

- Initialize 80% of main memory to random data
- Write 80% of memory to disk

## Read test

- Read back file written in previous test

$$R_{PEAK} = \frac{2(0.8S_{MEM})}{T_{PREAD} + T_{PWRITE}}$$

# Peak Disk I/O Goals

## Tests optimized system level I/O performance

- Paging
- Checkpoint/Restart
- Highest "achievable" disk performance

## Tests ability to deal with *large* programs

- Uses 80% of system memory
- Difficult on some systems

## Application I/O Benchmark

**Emulates CFD applications that must write their solution fields over a set of iterations or time steps (e.g., for post-processing)**

**Derived from one of the NAS Parallel Benchmarks, BT**

- Simulates a pseudo-time stepping flow solver
- Solution method is a block tridiagonal matrix solve

**102x102x102x5 solution vector U is written every 5 out of 200 iterations in *serial* format (i.e., Fortran or C order of sequential bytes)**

## Application I/O Benchmark Goals

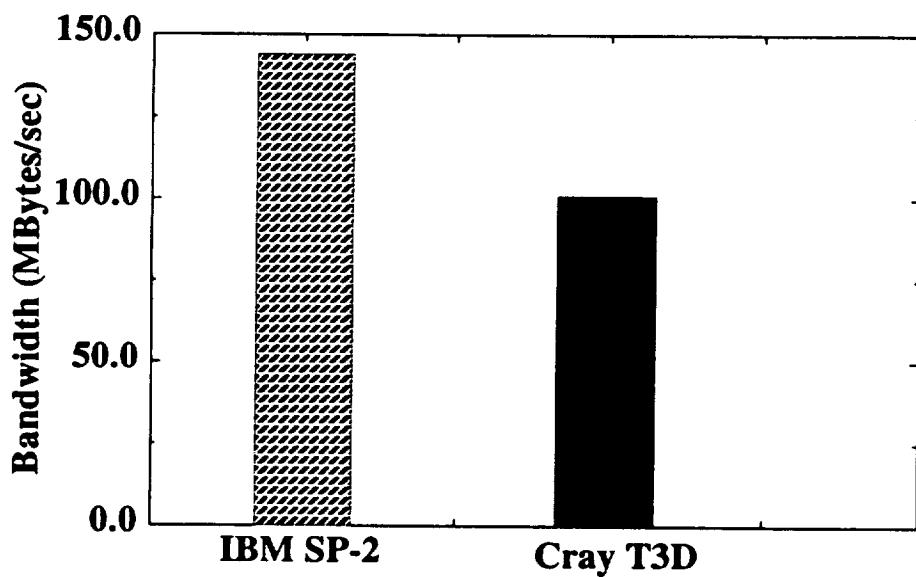
**Tests performance a “real” application can expect**

- Emulates CFD application *compute and I/O*
- Based on existing widespread benchmark, BT
- Allows system to use overlapped I/O

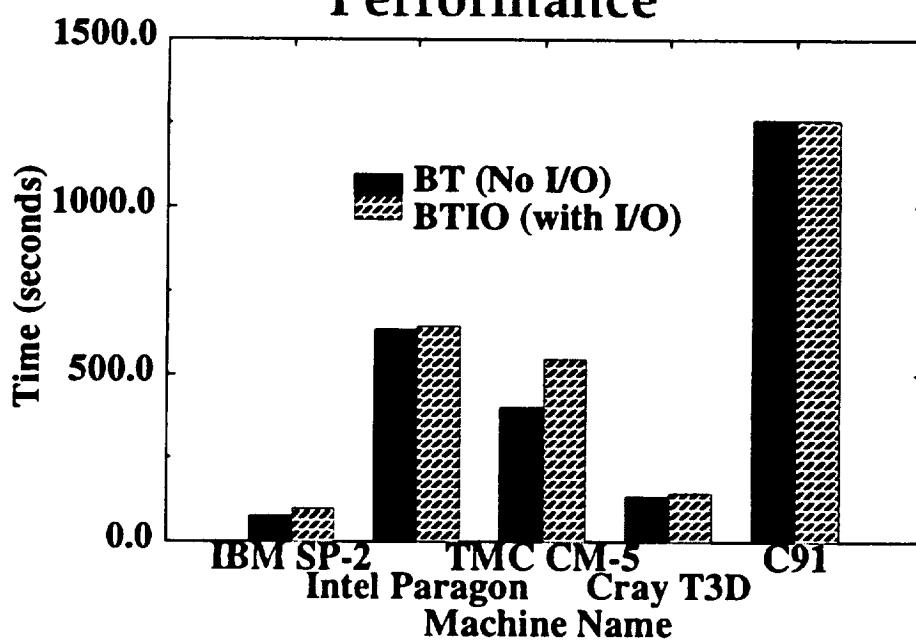
**Measures system “balance”**

- Involves significant amounts of both computation and disk I/O

## Peak I/O Benchmark Performance



## Application I/O Benchmark Performance



# System Descriptions

## Paragon

- 102 Compute Nodes, 32MB
- 32 I/O Nodes, 32MB
- Asynchronous write

## SP-2

- 64 nodes, 2 @ 512MB, 62 @ 128MB
- 1 disk/node

# System Descriptions (cont.)

## T3D (peak)

- 128 Processors, 1 I/O Gateway
- 2 DA60 disks (80 Mbyte/s peak, raid version of DD60)

## T3D (APPIO)

- 1024 processors, C98-256 front end
- 4 I/O Gateways
- 13 DD60 Disks (20MB/sec/disk)

# **Example Application Benchmark Implementations**

**iPSC/860**

**Paragon**

**IBM SP2**

# iPSC/860 Application I/O Benchmark

## Code used 3D decomposition

- good for compute, not I/O

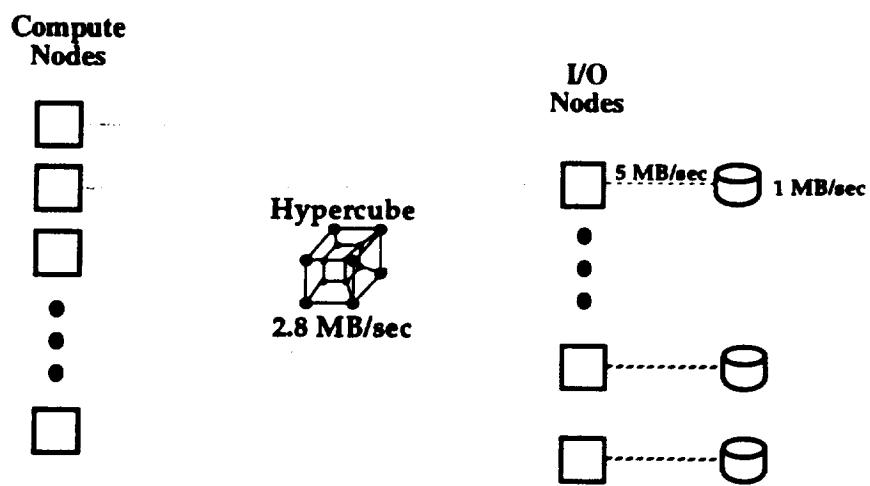
## Very small memory

- Data re-arrangement needed to be done on disk

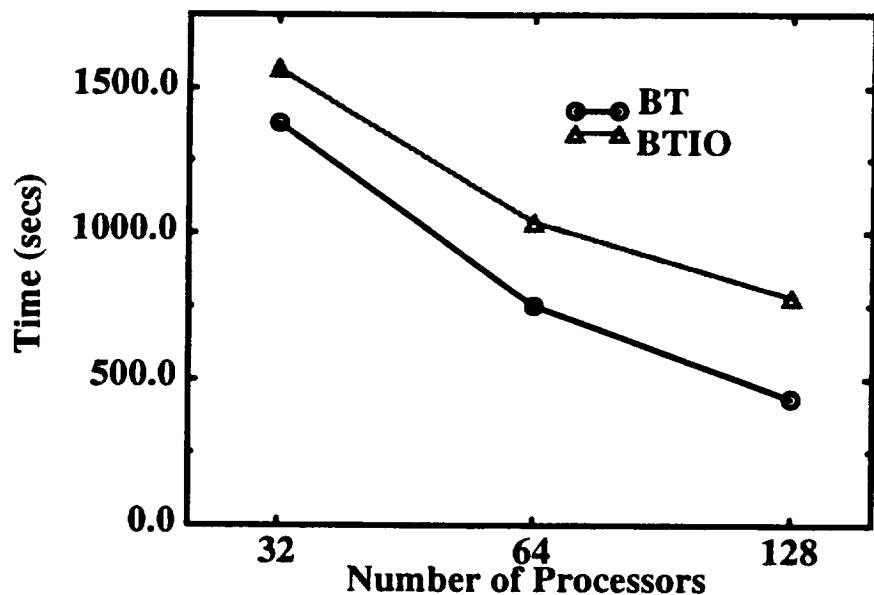
## Small blocks

- Overloaded I/O nodes
- Contention for disk cache space

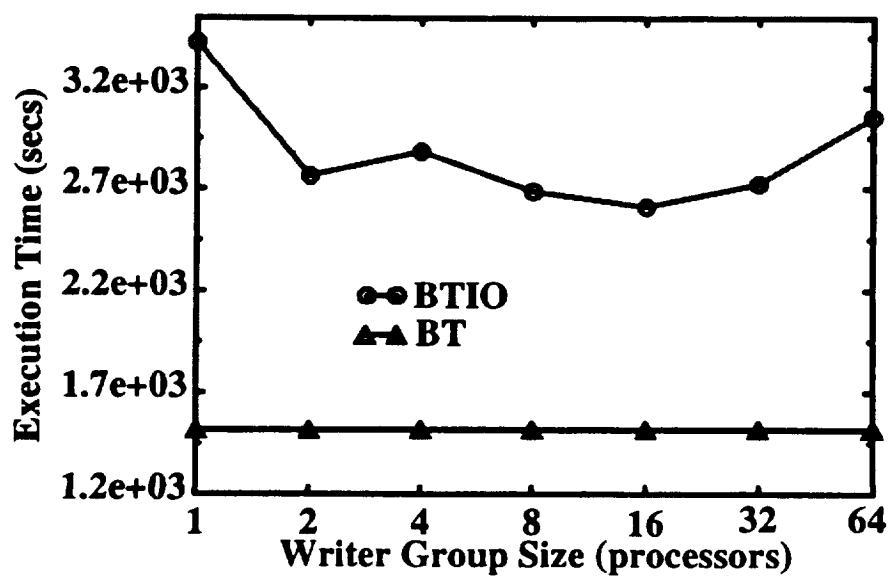
# Intel iPSC/860 I/O Architecture



## iPSC/860 Application I/O Benchmark Performance ( $64^3$ problem size)



## Effect of Varying Number of Writers on Execution Time ( $102^3$ size)



## *A Fast Implementation of the*

### *NAS Application I/O Benchmark*

*on*

### *the Intel Paragon Supercomputers*

- II. Implementing NAS CFD Application I/O code :*
- II.a The transposition method and data decomposition :*
- 1. Each compute nodes has one plane of grid data ,*
  - 2. Divide and conquer among nodes using the third dimension ,*
  - 3. Overlap RHS communication with computing tasks ,*
  - 4. Fast all-to-all communication for the LHS ,*

#### *II.b I/O requirements and file size for class B data size :*

- 1. Compute nodes perform the write and save the output using Paragon Parallel File (PFS) with M\_RECORD mode ,*
- 2. Each write size is the entire plane of data ~ 416 Kbytes/Node ,*
- 3. Total size of the output file is 1.7 GBytes ,*

*Thanh Phung*

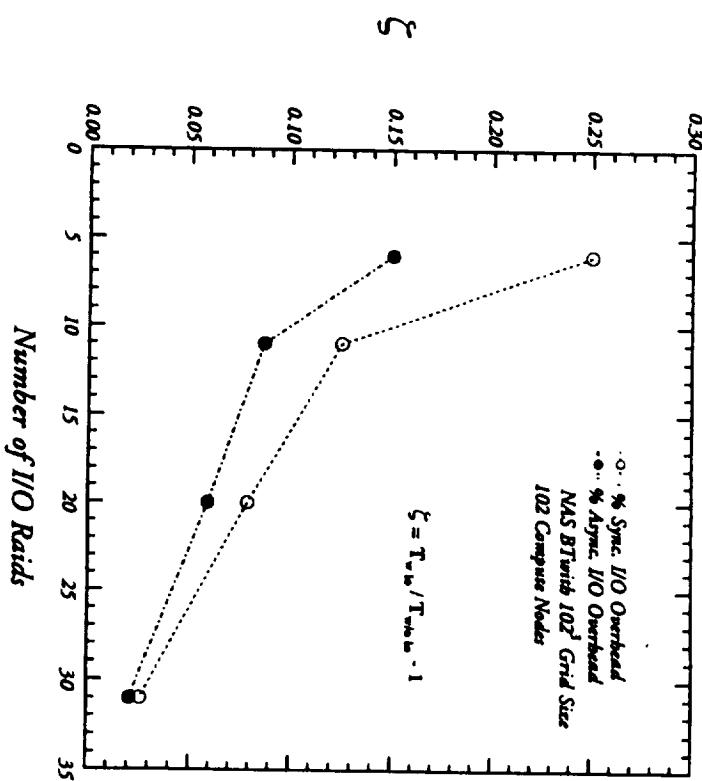
*Intel Supercomputer System Div.*

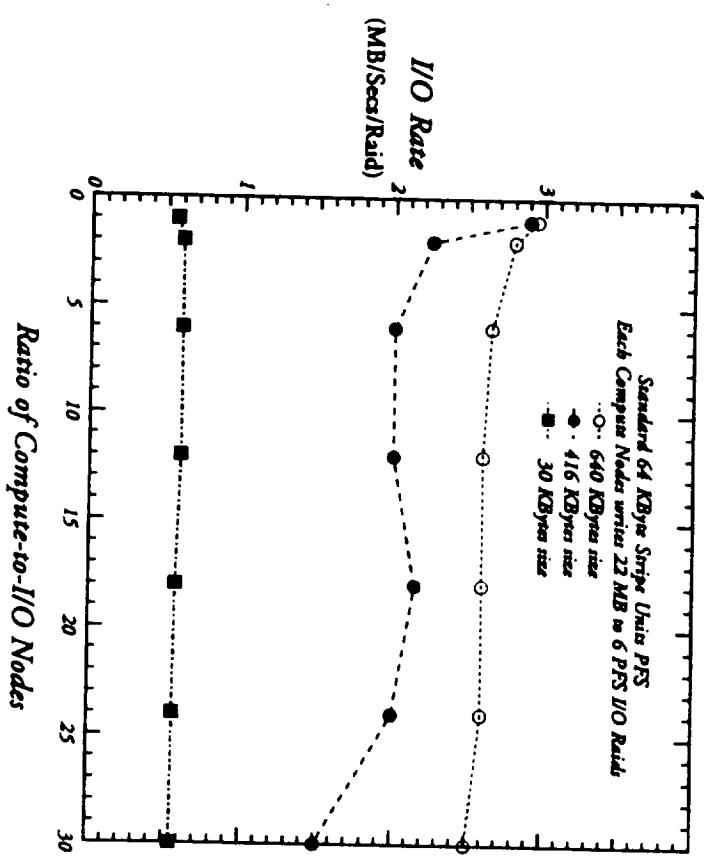
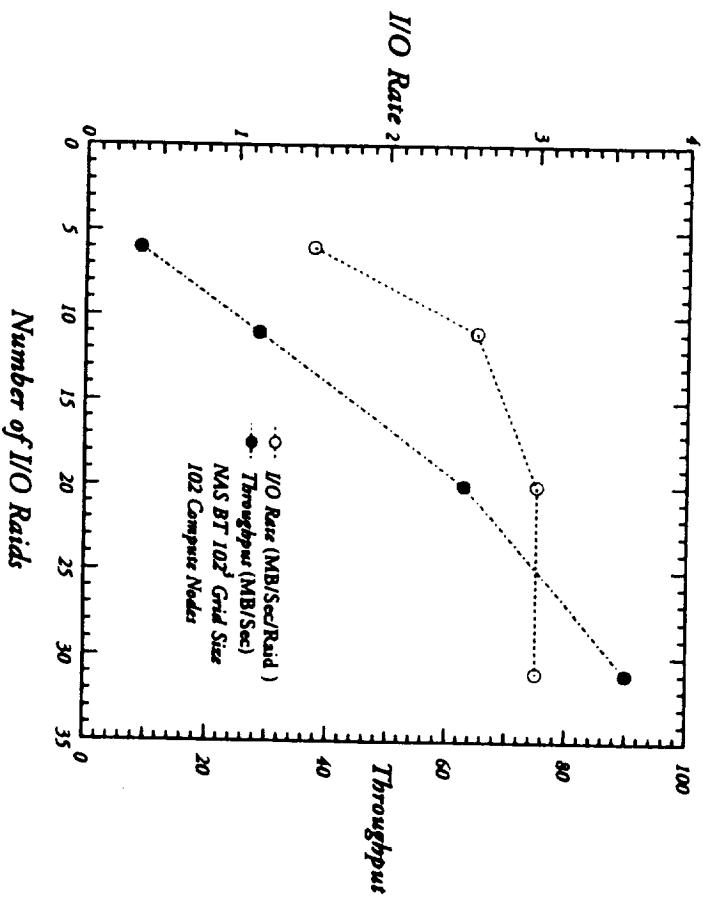
*NASA Ames Research Center*

*Moffett Field , CA*

III. Obtaining fast I/O on the Paragon supercomputers:

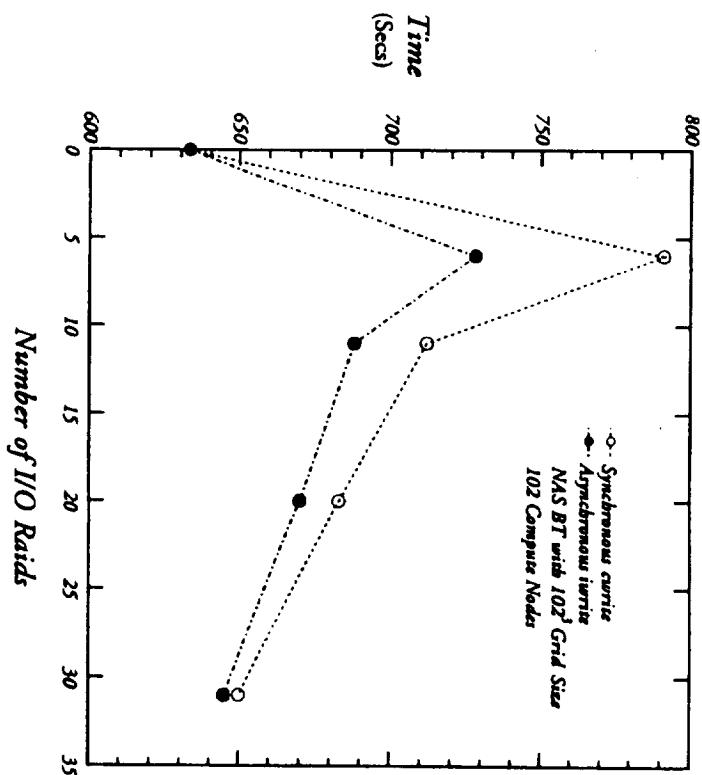
1. Schedule I/O tasks to overlap with both computation and more importantly inter-processor communication,
2. Use PFS with M\_RECORD mode to obtain best I/O performance with large write size 416 Kbytes per node,
3. Perform asynchronous PFS iowrite to avoid I/O blocking on the compute nodes,
4. The compute-to-I/O raid ratio should be 10:1 or less.





#### IV Conclusions:

1. Good PFS performance is obtained with I/O rate of closed to 3.0 MB/Sec/Raid,
2. Excellent total throughput rate as total number of PFS raids increase,
3. Fast parallel I/O can be obtained by:
  - . overlapping I/O tasks with not only computation but also internode communication,
  - . utilized the asynchronous PFS I/O to avoid I/O blocking.
  - . read/write size is preferably as large as the default stripe unit,
  - . compute-to-I/O raid ratio should be kept less than 10 to 1.



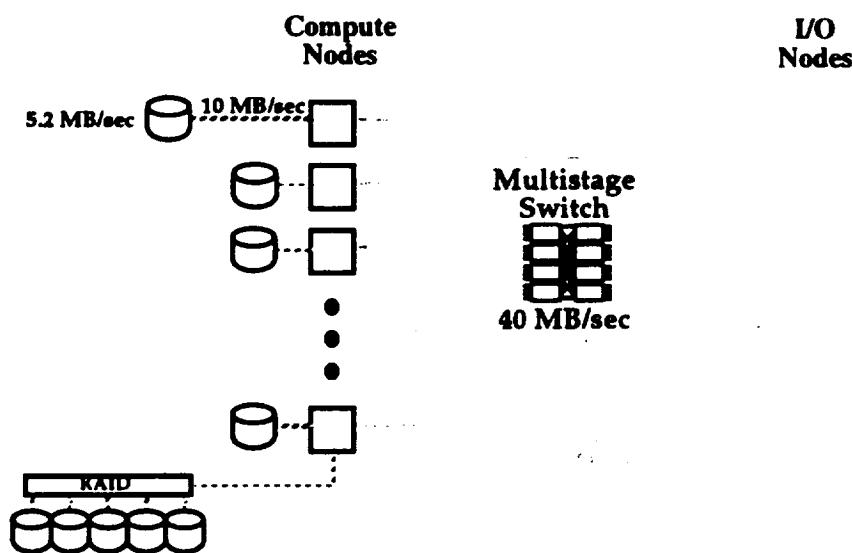
# SP1/SP2 Application I/O Benchmark

Work performed by Vijay K. Naik, IBM

## Implementation

- Computation portion same as BT
- Solution vector saved every 5 iterations
- Data re-distributed after BT finishes
- 40 solution vectors to 40 files
- I/O on 40 local disks

## IBM SP1/SP2 I/O Architecture



# Performance Considerations

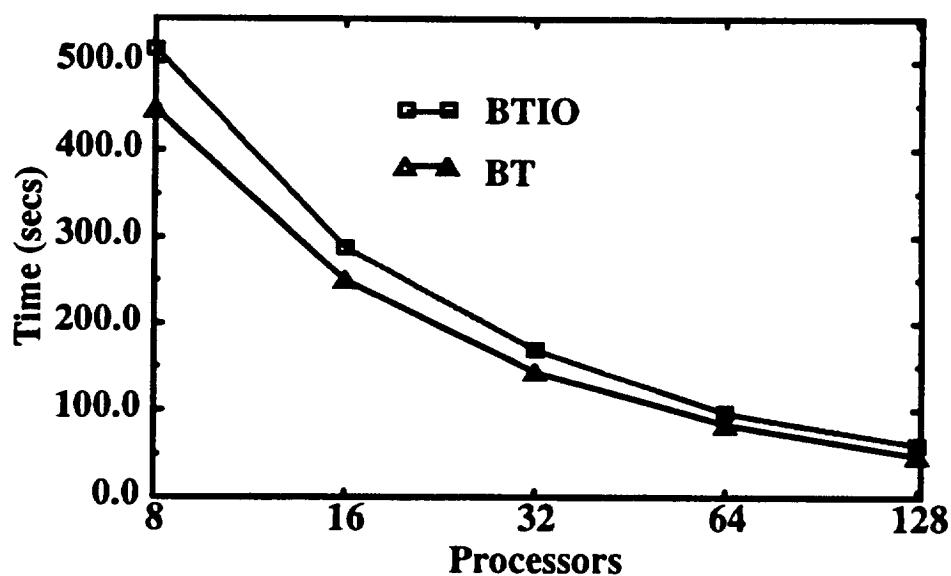
## Takes advantage of large per-node memory

- Data re-arrangement into contiguous chunks
- Allows parallelism over non-parallel file system

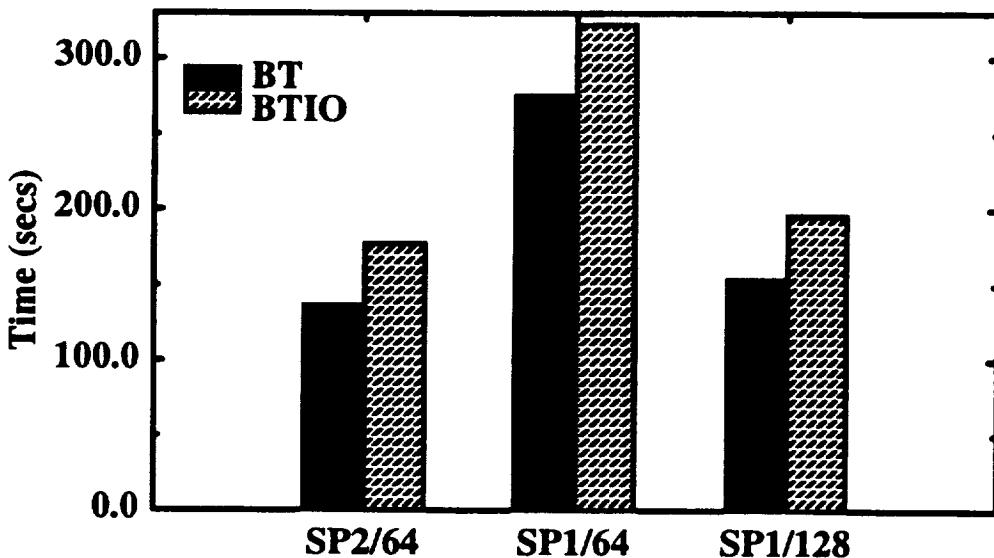
## Disadvantages

- Would not be possible if application was larger
- I/O limited to at most 40 I/O nodes
- Does not use asynchronous I/O

## SP1 Class A Application I/O Performance



## Class B SP2 Application I/O Performance



## Benchmarks Summary

NHT-1 I/O benchmarks are able to

- Compare system performance
- Demonstrate differences between coding techniques

### Limitations

- Applications must reflect desired workload
- Some optimizations not possible with some applications
- System I/O configurations vary widely

# I/O Libraries and User Interfaces

SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994



## Types of Parallel I/O Interfaces

### Unix like

- UNIX read/write/seek
- Workstations/clusters/Cray C90

### UNIX with I/O modes

- UNIX read/write/seek
- Modes specify file pointer semantics
- Parallel functionality in system software
- PFS, CFS, CM5 SDA



SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

## Types of Parallel I/O Interfaces (cont.)

### Library based

- Special I/O library
- Must use library to "see" parallel file system
- IBM VESTA, MPI-IO, MasPar?

## I/O Access Methods

### Independent

- Processors access files independently
- + Less synchronization
- Does not allow in-memory data re-arrangement, may overload I/O servers

### Collective

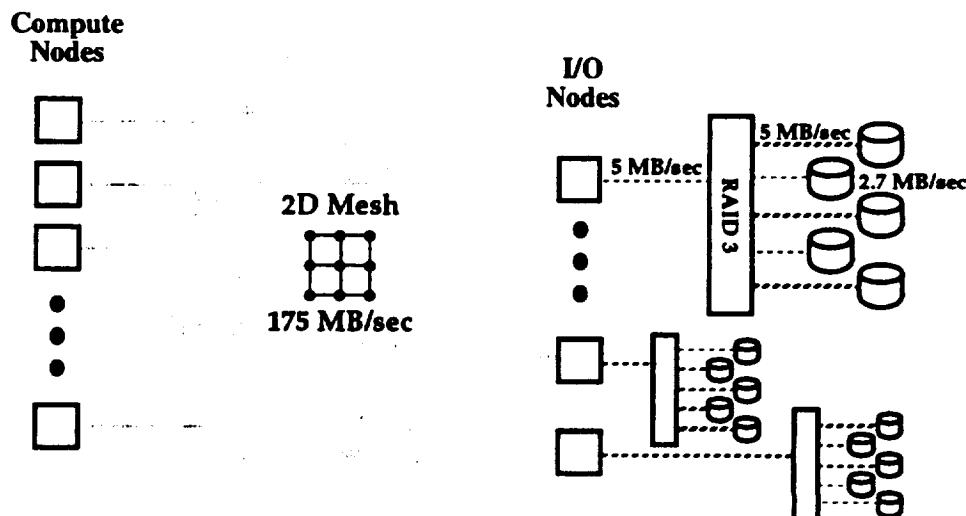
- All processors access files together
- + Can use in-memory re-arrangement, load to I/O server can be restricted
- Requires synchronization

# Intel PFS (Paragon)

SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

## Intel Paragon I/O Architecture



# **Paragon**

## **Parallel File System**

Brad Rullman

Intel Supercomputer Systems Division

9/7/94

1

## **OSF/1 Compatibility**

- **PFS system calls work on standard OSF/1 files**
- **Standard OSF/1 I/O system calls work on PFS files**
- **Standard OSF/1 commands work on PFS files**
  - A subset have been modified to support files >2G-1 bytes in size
- **PFS supports standard OSF/1 file permission semantics**

Brad Rullman

Intel Supercomputer Systems Division

9/7/94

•

# File Sharing Modes

- **setiomode()** sets/resets the I/O mode of a file
- All file systems supported (PFS, UFS, NFS)
- **M\_UNIX mode**
  - Each node has a unique file pointer
  - Nodes are not synchronized
  - Variable-length, unordered records
- **M\_LOG mode**
  - Shared file pointer
  - Nodes are not synchronized
  - Variable-length, unordered records

Brad Rullman

Intel Supercomputer Systems Division

9/7/94

10

# File Sharing Modes (cont.)

- **M\_SYNC mode**
  - Shared file pointer
  - Nodes are synchronized
  - Variable-length records, stored in node order
- **M\_RECORD mode**
  - Unique file pointer
  - Nodes are not synchronized
  - Fixed-length records, stored in node order
  - Highly parallel

Brad Rullman

Intel Supercomputer Systems Division

9/7/94

11

# File Sharing Modes (cont.)

- **M\_GLOBAL mode**
  - Shared file pointer
  - Nodes are synchronized
  - Variable-length, unordered records
  - All nodes access the same data
  - Data read/written from/to disk only once
- **M\_ASYNC mode (R1.3)**
  - Unique file pointer
  - Nodes are not synchronized
  - Variable-length, unordered records
  - Multiple readers/multiple writers are allowed with no restrictions

Brad Rullman

Intel Supercomputer Systems Division

9/7/94

12

## Global Open

- Performs a global open operation (only 1 open request is issued, rather than  $n$  requests)
- Sets the I/O mode of the file ( $n$  separate setiomode() calls are not necessary)
- Synchronizing call (all nodes in the application must call it)

Brad Rullman

Intel Supercomputer Systems Division

9/7/94

11

# Support for Large (>2G-1 bytes) Files

- “Extended” interfaces support files <  $2^{63}-1$  bytes
- **Extended seek**
  - `esseek()` — syntax mirrors OSF/1 `Iseek()`
- **Extended size for disk block preallocation**
  - `Isize()` / `esize()` — syntax mirrors `Iseek()` / `esseek()`
  - Also allows pre-existing files to be extended
- **Extended stat**
  - `estat()` / `festat()` / `Iestat()` — syntax mirrors `stat()` / `fstat()` / `Istat()`

Brad Rullman

Intel Supercomputer Systems Division

9/7/94

14

## Other Features

- **Asynchronous I/O**
- **PFS file data is not cached for reuse**
  - UFS file data is cached on the compute node
- **Uses “Fast Path” data access**
  - Streamlined path between user’s data buffer and disk storage
  - Avoids extra data copies required for caching

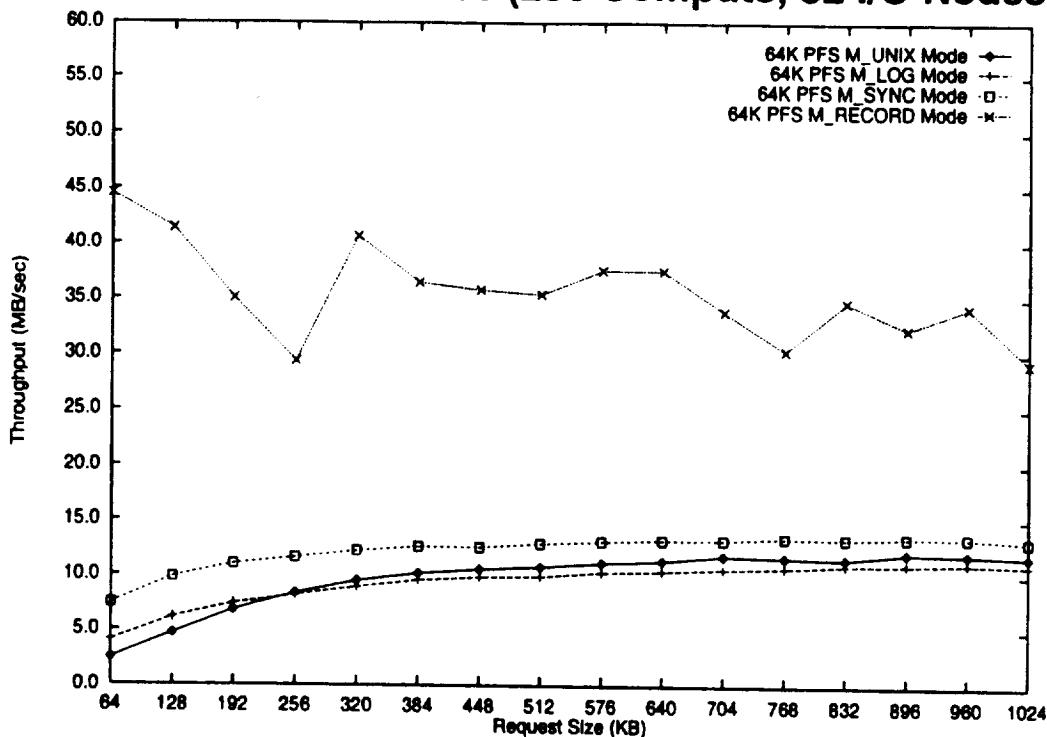
Brad Rullman

Intel Supercomputer Systems Division

9/7/94

15

## PFS Write Performance (256 Compute, 32 I/O Nodes)



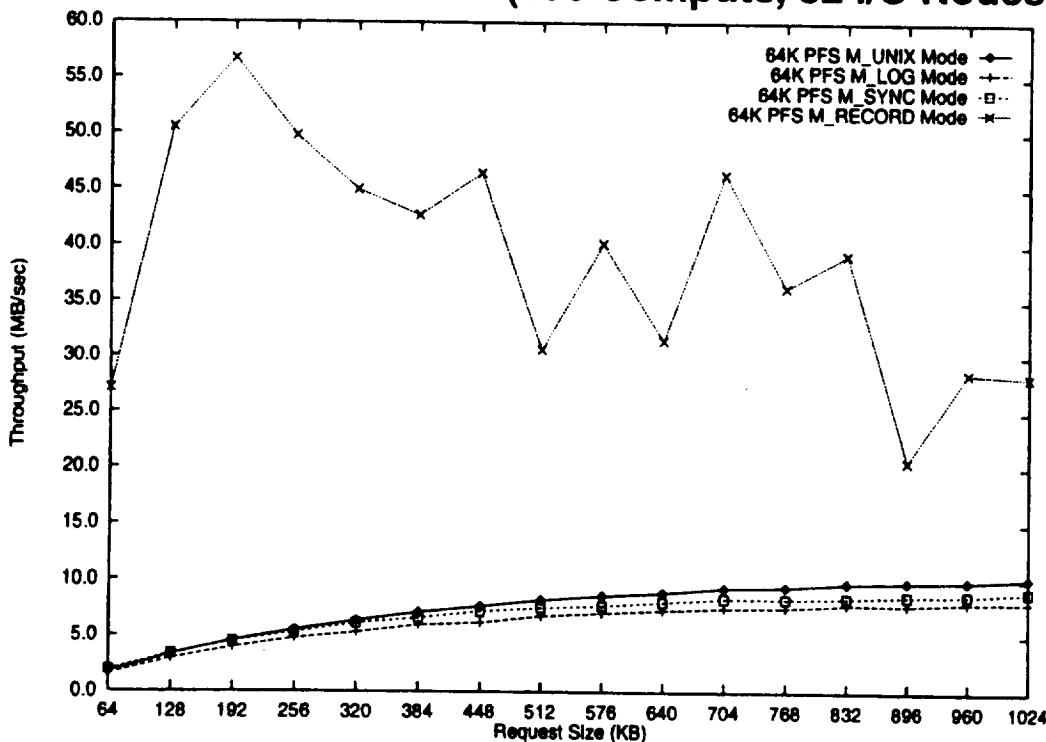
Brad Rullman

Intel Supercomputer Systems Division

6/24/94

22

## PFS Read Performance (256 Compute, 32 I/O Nodes)



Brad Rullman

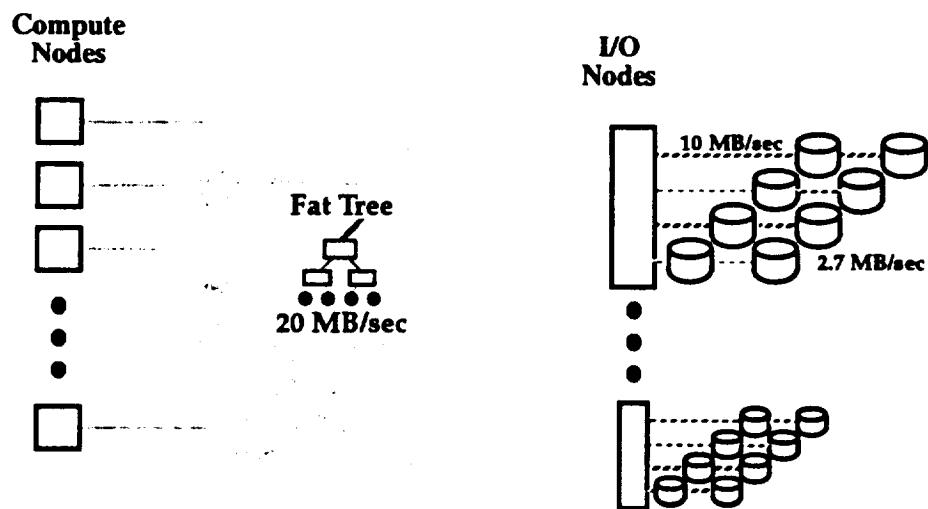
Intel Supercomputer Systems Division

6/24/94

23

# CM5 SDA

## TMC CM-5 I/O Architecture



# CM5 Programming Models

## CM FORTRAN

- HPF-like, Emulates SIMD
- Parallelism through array operations
- Data distributed through compiler directives

## CMM

- Normal message passing

# CM FORTRAN I/O

**Normal FORTRAN open/read/write**

**All I/O is collective**

**Data transposition done in RAM as part of read/write operation**



# CMMMD I/O

## Supports I/O modes

- Local:  
CMMMD\_local
- Global:  
CMMMD\_independent  
CMMMD\_sync\_bc  
CMMMD\_sync\_seq - synchronous sequential

## Independent I/O operations are serialized

- Must use collective to go fast - CP bottleneck

# CMMMD I/O (cont.)

## CMMMD\_sync\_bc

	Node 0 0-511	Node 1 0-511	Node 2 0-511	

## CMMMD\_sync\_seq

	Node 0 0-511	Node 1 511-1023	Node 2 1023-1535	

# Support for BIG Files

**Unix uses 32-bit file pointer**

- files can't be larger than 2Gb

**To get around this limitation**

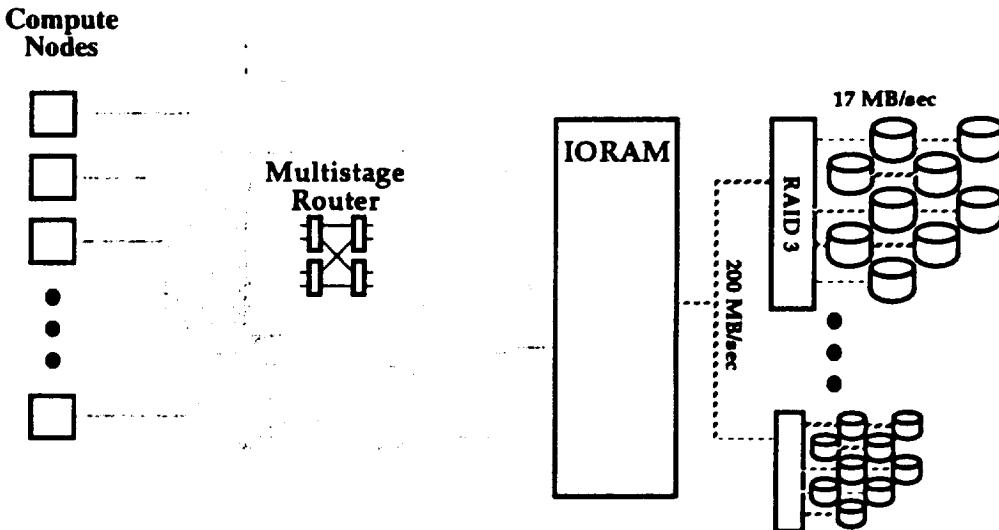
- CMMD provides the usual functions with file positions encoded as double precision floating-point numbers
- Can't do this in "global" CM Fortran, but there the support should be *transparent* to the user



# MasPar I/O



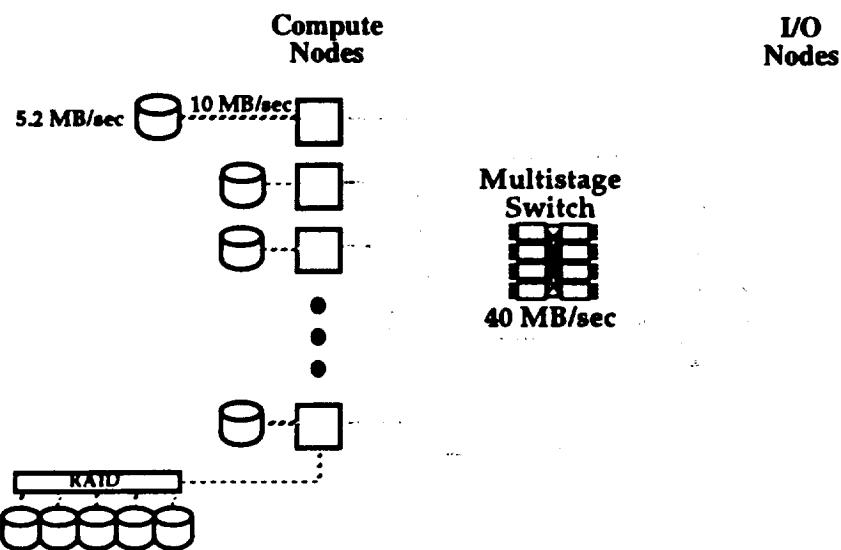
## MasPar MP-2 I/O Architecture



SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

## IBM SP2 I/O Architecture



SC '94 Parallel I/O Tutorial, B. Nitzberg and S. Fineberg

Nov. 14, 1994

# **The MasPar Scalable Unix I/O System**

---

*International Parallel Processing Symposium 1994*

**John R. Nickolls**

**MasPar Computer Corporation**  
**749 N. Mary Avenue**  
**Sunnyvale, CA 94022**  
**[nickolls@maspar.com](mailto:nickolls@maspar.com)**

---

**MasPar Computer Corporation**  
An SGI™ MasPar I/O System

**MASPAR**

## **Unix Data-Parallel I/O Model**

---

*Minimal extensions to Unix I/O model*

- Conventional Unix file descriptors
- Intermixed scalar I/O and parallel I/O
- Conventional scalar Unix I/O
  - open, close, lseek, read, write, ioctl
- Data parallel I/O functions for MPL and MPF
  - Aggregate sets
  - Arrays
- Accommodates distributed memory
- Extensions for asynchronous I/O and I/O buffer management

---

**MasPar Computer Corporation**  
An SGI™ MasPar I/O System

**MASPAR**

# Parallel Active Set I/O

*Flexible parallel I/O for arbitrary data sets*

- MPL plural variables represent sets
- If statements with plural expressions select subsets
  - Enable corresponding active set of processors
- MPL example:

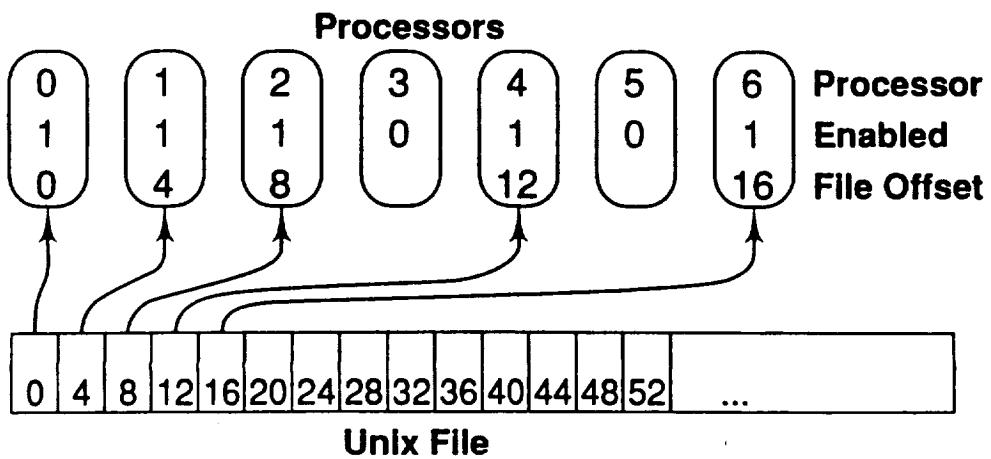
```
plural char p_buf[BUFSIZE];
fd = open("file", mode);
nread = p_read(fd, p_buf, nbytes);
```
- Enumeration scan of active processors determines file offset for each participating processor

**MASPAR**

MasPar Computer Corporation  
TM MPPS™ MasPar I/O System

# Interleaved Active Set Parallel I/O

`nread = p_read(fd, p_buf, 4);`



**MASPAR**

MasPar Computer Corporation  
TM MPPS™ MasPar I/O System

# Flexible Active Set I/O

## *I/O addressing autonomy*

- Active processors compute plural file positions `p_offset`
- Overlapping file offsets permitted
- MPL example:

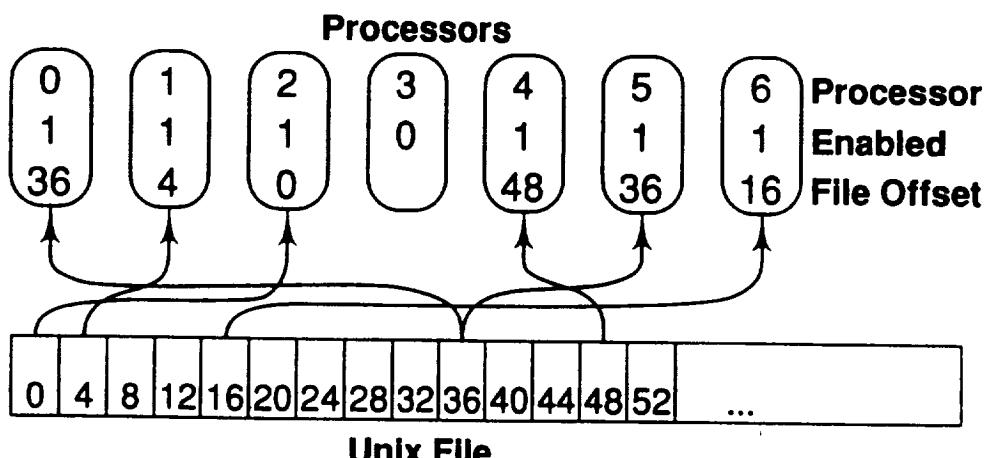
```
plural char p_buf[BUFSIZE];
plural off_t p_offset;
pp_lseek(fd, p_offset, L_SET);
nread = pp_read(fd, p_buf, nbytes);
```
- File offsets typically data-dependent or coordinated permutations of processor numbers

**MASPAR**

MasPar Computer Corporation  
PP-PPV14 MasPar I/O System

# Flexible Active Set Parallel I/O

```
pp_lseek(fd, p_offset, L_SET);
nread = pp_read(fd, p_buf, 4);
```



**MASPAR**

MasPar Computer Corporation  
PP-PPV14 MasPar I/O System

# Parallel Array I/O

*Arrays are the primary parallel data structure*

- 1-D, 2-D, n-D arrays
- Block and cyclic processor data mappings
- MPL example:

```
plural short image[nrow/nyproc][ncol/nxproc];
nread = read2dh(fd, image, nrow, ncol, 2);
```

- MPF Fortran 90 unformatted I/O example:  

```
real, dimension (1000, 2000) :: A, B
open (unit=10,file='out',form='unformatted')
write (unit=10) A, B(:, 1)
```

- Fortran record header, data in array element order, trailer

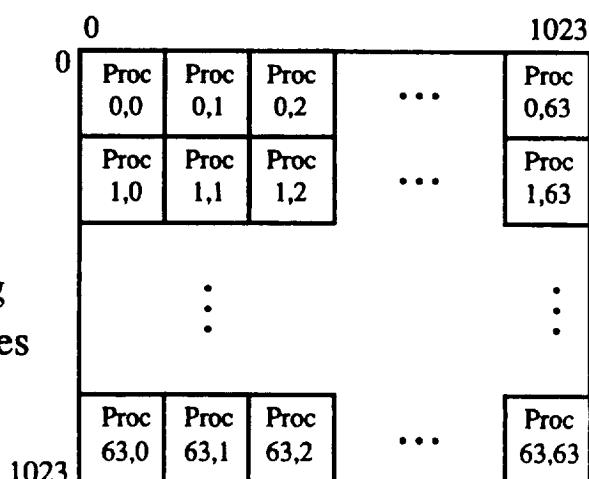
**MASPAR**

MasPar Computer Corporation  
MS-PPS '94 MasPar I/O Systems

# Parallel Block Mapped 2-D Array I/O

```
nread=read2dh(fd,image,nrow,ncol,2);
```

- Example:  
1024 x 1024  
matrix on a  
64 x 64  
processor grid
- 2-D Block mapping
- 16 x 16 Sub-matrices



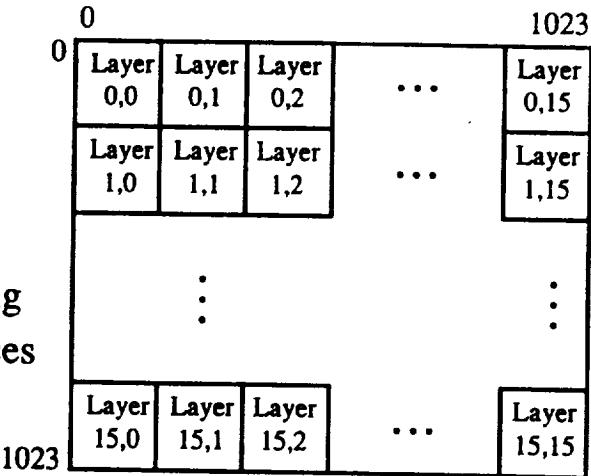
**MASPAR**

MasPar Computer Corporation  
MS-PPS '94 MasPar I/O Systems

# Parallel Cyclic Mapped 2-D Array I/O

```
nread=read2dc(fd,image,nrow,ncol,2);
```

- Example:  
1024 x 1024  
matrix on a  
64 x 64  
processor grid
- 2-D Cyclic mapping
- 64 x 64 Sub-matrices
- 16 x 16 Layers



**MASPAR**

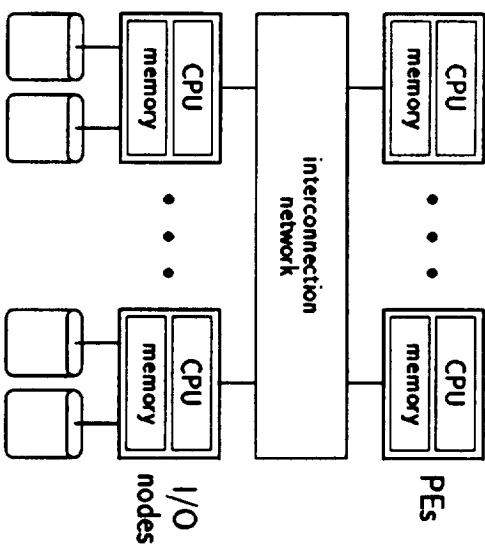
MasPar Computer Corporation  
An IBM Company  
MasPar I/O Systems

## The Vesta Parallel File System

Peter F. Corbett      Dror G. Feitelson  
Sandra J. Baylor    Jean-Pierre Prost    Yasun Hsu

T J Watson Research Center  
Yorktown Heights, NY 10598

## Vesta Parallel I/O Model



### Outline:

1. Parallel I/O
2. Vesta concepts
3. Parallel access to Vesta files
4. Implementation
5. Performance

- ▷ Scalable bandwidth and capacity
- ▷ Low latency
- ▷ Staging area for archival mass storage systems

## Vesta Design Principles

Parallelism:

- ⇒ Define a new parallel user interface
- ⇒ Preserve parallelism from the application to the disks

Scalability:

- ⇒ Eliminate centralized control points
- ⇒ Support direct access to data

Layering:

- ⇒ Based on reliable services provided by lower layers
- ⇒ Provide basic abstractions and services for parallel I/O
- ⇒ High-level services left for higher layers, e.g. libraries with collective I/O

Superset of common features:

- ⇒ Break 2GB file size limit
- ⇒ Support asynchronous I/O and prefetching
- ⇒ File checkpointing
- ⇒ Hierarchical directory structure

## Vesta Key Concept

New abstraction: **a file is multiple sequences of bytes.**

Files can be partitioned

- » Each PE opens a partition rather than the whole file
- » Partitions are disjoint
- » PEs enjoy asynchronous parallel access to different partitions
- » Various partitioning schemes are supported

The price: a new nonstandard user interface.

## Examples of Partitioning Schemes

### Partitioning Vesta Files

Cells:

- ▷ Files have a 2-D structure, defined in terms of cells
- ▷ Cells are like virtual disks
- ▷ Cells are mapped to I/O nodes in round-robin manner
- ▷ Defined when file is created

Subfiles:

- ▷ Different views of the data are possible without actually moving the data
- ▷ Common rectilinear decompositions are supported
- ▷ Defined by four parameters:
  - $Vbs$ : vertical block size, in multiples of the basic striping unit
  - $Hbs$ : horizontal block size (number of cells spanned)
  - $Vn$ : number of subfiles in vertical dimension
  - $Hn$ : number of subfiles in horizontal dimension
- ▷ Defined when file is opened

$Vbs = 1$	$0$	$1$	$2$	$3$	$0$	$1$	$2$	$3$	$0$	$4$	$5$	$6$	$7$
$Vn = 1$	$4$	$4$	$4$	$4$	$4$	$4$	$4$	$4$	$4$	$4$	$4$	$4$	$4$
$Hbs = 1$	$5$	$5$	$5$	$5$	$5$	$5$	$5$	$5$	$5$	$5$	$5$	$5$	$5$
$Hn = 4$	$6$	$6$	$6$	$6$	$6$	$6$	$6$	$6$	$6$	$6$	$6$	$6$	$6$
	$7$	$7$	$7$	$7$	$7$	$7$	$7$	$7$	$7$	$7$	$7$	$7$	$7$

$Vbs = 1$	$0$	$1$	$2$	$3$	$0$	$1$	$2$	$3$	$0$	$4$	$5$	$6$	$7$
$Vn = 4$	$0$	$1$	$2$	$3$	$0$	$1$	$2$	$3$	$0$	$1$	$2$	$3$	$4$
$Hbs = 4$	$4$	$5$	$6$	$7$	$4$	$5$	$6$	$7$	$4$	$5$	$6$	$7$	$4$
$Hn = 1$	$4$	$5$	$6$	$7$	$4$	$5$	$6$	$7$	$4$	$5$	$6$	$7$	$4$

$Vbs = 2$	$1$	$3$	$5$	$7$	$0$	$2$	$4$	$6$	$1$	$3$	$5$	$7$	$0$
$Vn = 4$	$0$	$2$	$4$	$6$	$0$	$2$	$4$	$6$	$0$	$2$	$4$	$6$	$0$
$Hbs = 4$	$12$	$14$	$16$	$18$	$12$	$14$	$16$	$18$	$12$	$14$	$16$	$18$	$12$
$Hn = 1$	$3$	$5$	$7$	$9$	$3$	$5$	$7$	$9$	$3$	$5$	$7$	$9$	$3$

## Data Sharing Patterns

### Disjoint subfiles

- No synchronization or concurrency control

### Shared subfiles

- No problems if read-shared
- Support for shared offsets
  - Asynchronous access with no prior coordination
  - Read successive records to distinct processes
  - Write successive records from distinct processes
- Concurrency control for shared writing
  - Sequential consistency: interleaved order consistent with each process
  - Linearizability: order consistent with inter-process synchronization
  - Atomicity: same order at all I/O nodes

## File Usage Scenario

### 1. Vesta Create

Specify number of cells, basic striping unit, I/O nodes used, and disk pre-allocation.

Done once.

### 2. Vesta\_Attach

Authenticate access rights, get layout information.  
Limited to one writing job at a time.

Done once from every compute node that requires access.

### 3. Vesta\_Open

Set the partitioning parameters and subfile.  
Multiple views can be used on each compute node.

### 4. Vesta Read and Vesta Write

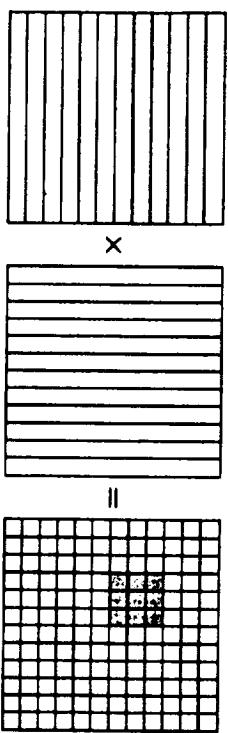
Access data in the specified subfile.  
Asynchronous version allows overlap with computation.

### 5. Vesta Close

### 6. Vesta Detach

Release the file.

## Example: Matrix Multiplication



### Implementation

#### Basic features

- Client-server structure
- Client is linked to application on compute node.
- All knowledge of partitioning is confined to client.
- Server runs on I/O nodes.
- All knowledge of disk layout, and all concurrency control, are confined to server.

- Server maintains buffer cache and implements prefetching and write-behind
- Metadata is distributed and found by hashing object name

#### Environment

- IBM SP1 with EUI-H
  - Server uses AIIX logical volume layer for actual disk access
  - Each node has IBM 0663 hard disk
    - ⇒ 1.5 MB/s for write
    - ⇒ 2.2 MB/s for read
  - Integrated into experimental runtime system
    - ⇒ Server is a separate persistent partition
    - ⇒ Support for inter-partition communication
- ```

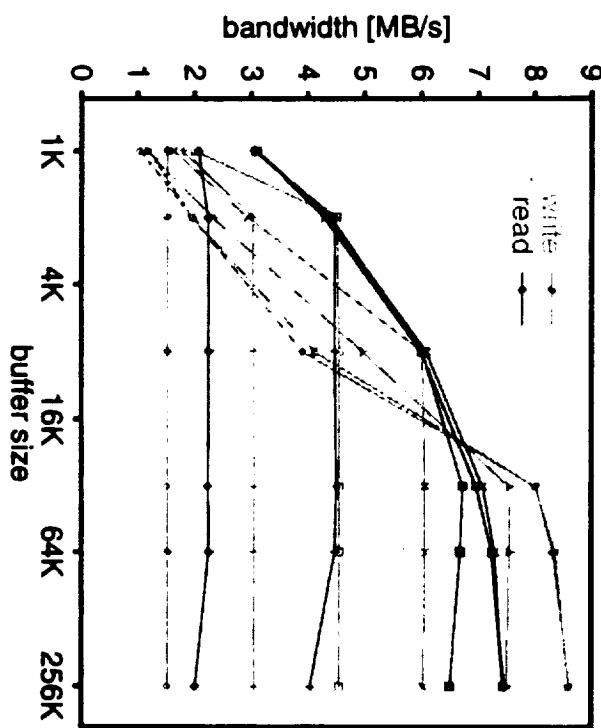
/*
 * parameters:
 *   id is process id
 *   N is matrix size
 *   P is number of processors
 *   sqrtP is sqrt(P)
 *   blk is N / sqrtP */
int fda, fdb, fdc;
float A[N][blk], B[blk][N], C[blk][blk];
/* open horizontal band across matrix A */
fda = Vesta_Open( "matA", blk, sqrtP, N, 1, id/sqrtP );
/* open vertical band across matrix B */
fdb = Vesta_Open( "matB", N, 1, blk, sqrtP, id/sqrtP );
/* open intersection in matrix C */
fdc = Vesta_Open( "matC", blk, sqrtP, blk, sqrtP, id );
/* read input data */
Vesta_Read( fda, A, N*blk );
Vesta_Read( fdb, B, N*blk );
/* perform the multiplication */
for (i=0 ; i<blk ; i++) {
  for (j=0 ; j<blk ; j++) {
    for (k=0 ; k<N ; k++)
      C[i][j] += A[i][k] * B[k][j];
  }
}
/* write the result */
Vesta_Write( fdc, C, blk*blk );

```

## Scaling I/O Nodes

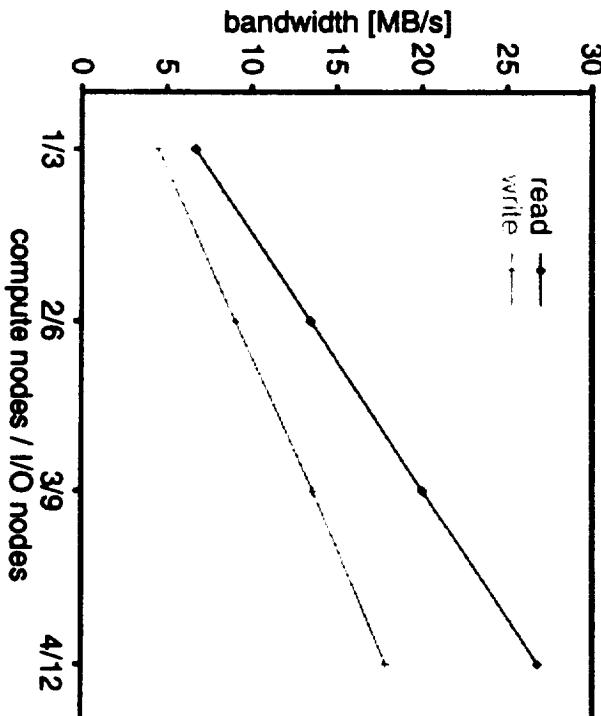
Measurements on IBM SP1

Using 1 compute node and multiple I/O nodes



## Scaling the Whole System

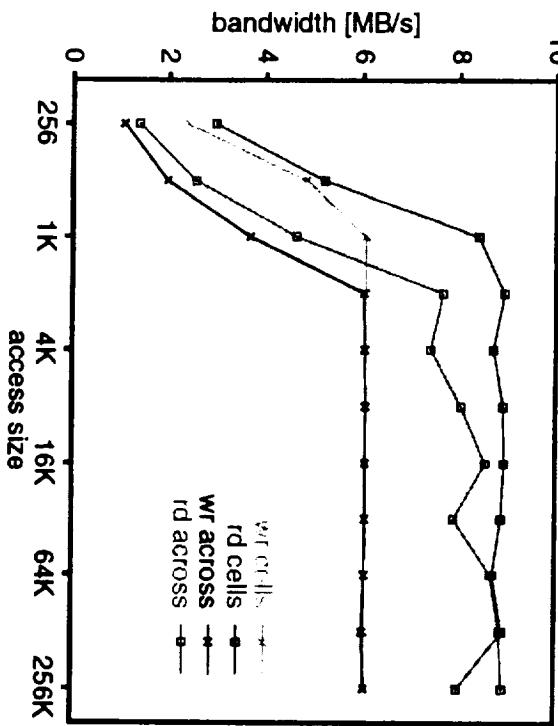
Measurements on IBM SP1  
Constant ratio of 3 I/O nodes to each compute node



- Bandwidth is linear with number of I/O nodes
- Single compute node saturates at >7 MB/s for read
- Single compute node saturates at >8 MB/s for write
- Aggregate bandwidth is linear with system size

## Orthogonal Logical Views

Measurements on IBM SP1  
Using 4 compute nodes and 4 I/O nodes  
BSU is 1/4 of access size



Vesta provides...

- parallel access to files leads to adequate bandwidth for parallel applications
- partitioning and re-partitioning in various ways match layout with access patterns

- sharing among processes with concurrency control if needed
  - support for high-level interfaces e.g. collective I/O operations and transparent striping
- Additional features, including import/export to external storage, checkpointing on-the-fly, and prefetching data for sequential reads.

Status

- Stripping achieves bandwidth similar to independent access with smaller size
- Aggregate bandwidth matches disk bandwidth
- Prefetching and write-behind prevent disk scheduling problems

## Vesta Summary

# Parallel I/O Cheat Sheet

## **Write big blocks to disk**

- Transpose in memory not on disk

## **Overlap I/O with computation**

- Asynchronous I/O
- Must have memory to double buffer

## **Scaling**

- Reduce compute node to I/O node ratio

## **Use Collective I/O (only when it helps)**

# Futures...

## A higher-level parallel I/O interface

- Languages (e.g. HPF)?—maybe someday...
- I/O Libraries (e.g. MPI-IO)?—stay tuned...

## Better parallel file systems

- Disk-directed I/O (Kotz, Dartmouth, 1994)

## Workload support

## Attitude adjustment

**The processor is not the center of the Universe.**





## NAS TECHNICAL REPORT

Title:

Parallel I/O on Highly Parallel Systems  
Supercomputing '94 — Tutorial M11

Author(s):

Bill Nitzberg  
Samuel A. Fineberg

After approval:  
Design NAS  
Report number:

Branch Chief:

Approved: Thomas J. Wetherow

Date:

8 Dec 1994

NAS Report Number:

NAS-94-005